Viewing Transformations and Shading

Sagan Potenza

Code Structure

My code is an expanded upon version of the Model Transformations submission. It features the same Model class with a new function to calculate the normal of triangles/vertices. Additionally, the VBO has been modified to include a normal per vertex, either a vertex or triangle normal depending on settings.

The vertex and fragment shaders have been modified to include ambient/diffuse/specular shading, the ability to display the Z Buffer, and flat/gouraud/phong shading. This is done through if else statements and uniform variables which can be modified in the settings in the main file (see next section for list of settings).

Key Binds

Models can be controlled in a game like interface using the following keyboard controls:

Key	**Function**
Escape	Exits the program.
Up/Down Arrow	Translates the model along the x-axis.
Left/Right Arrow	Translates the model along the y-axis.
Page Up/Page Down	Translates the model along the z-axis.
R/T	Rotates the model around the x-axis.
Y/U	Rotates the model around the y-axis.
I/O	Rotates the model around the z-axis.
F/G	Scales the model along the x-axis.
H/J	Scales the model along the y-axis.
K/L	Scales the model along the z-axis.
Z/X	Decreases/Increases the FOV.

Settings

These are the settings available at the start of the main function that are relevant to the new functionality:

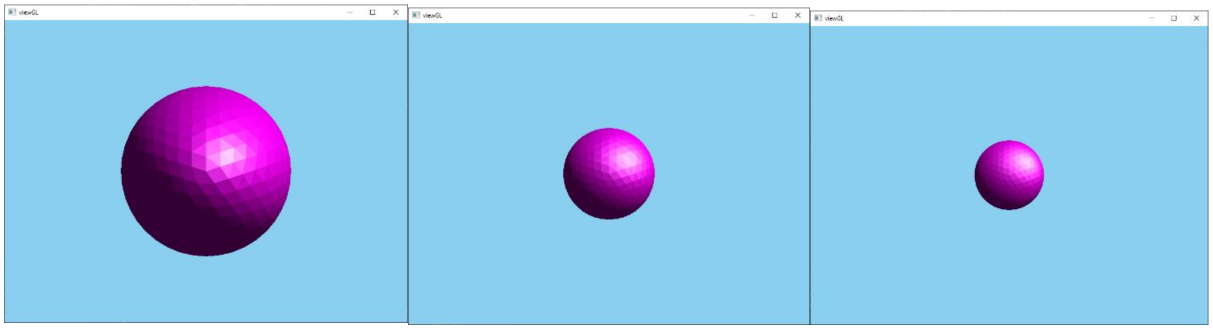
Variable	**Function**
zBuffer zBufferRenderMode	Either None, ZMode, ZTildeMode, or ZPrimeMode. If not None, renders the ZBuffer on screen.
shading shadingMode	Either None, Flat, Gouraud, or Phong. Renders that kind of shading.
float ambientLightIntensity	Ambient light intensity for ambient shading.
float lightIntensity	Light intensity for diffuse and specular shading.
float phongExponent	Phong exponent for specular shading.
glm::vec3 lightVec	Directional light direction for shading.
glm::vec3 specularColor	Color for specular shading.

Part 1: Perspective Transformation

A perspective transformation matrix is constructed via the glm::perspective function using the fov, near distance, far distance, and aspect ratio. Each of these can be changed to change how the object is viewed.

FOV

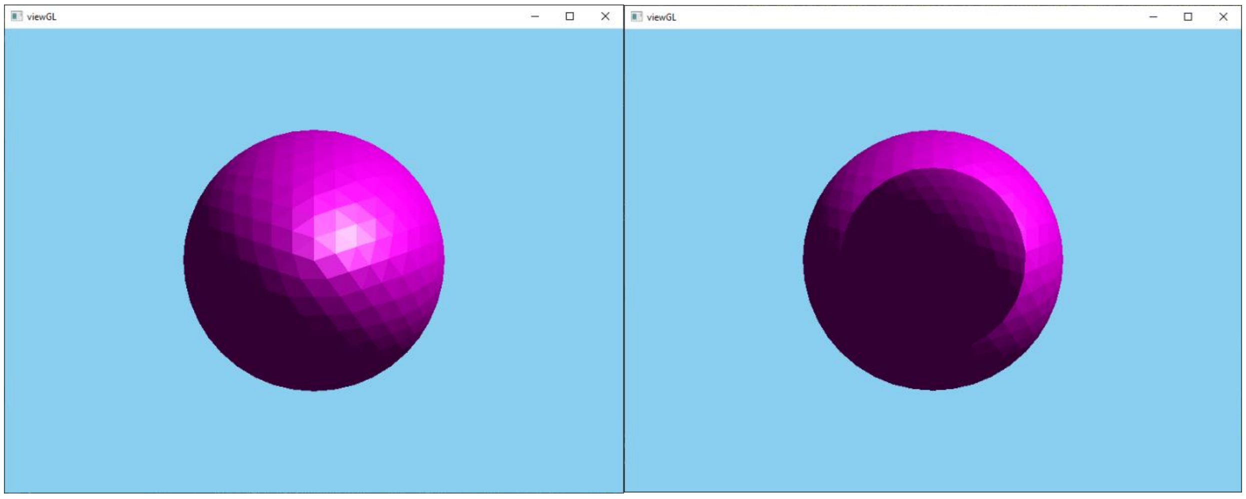
FOV can be varied to create a zoom like effect.



45° vs 75° vs 90°

Near Distance

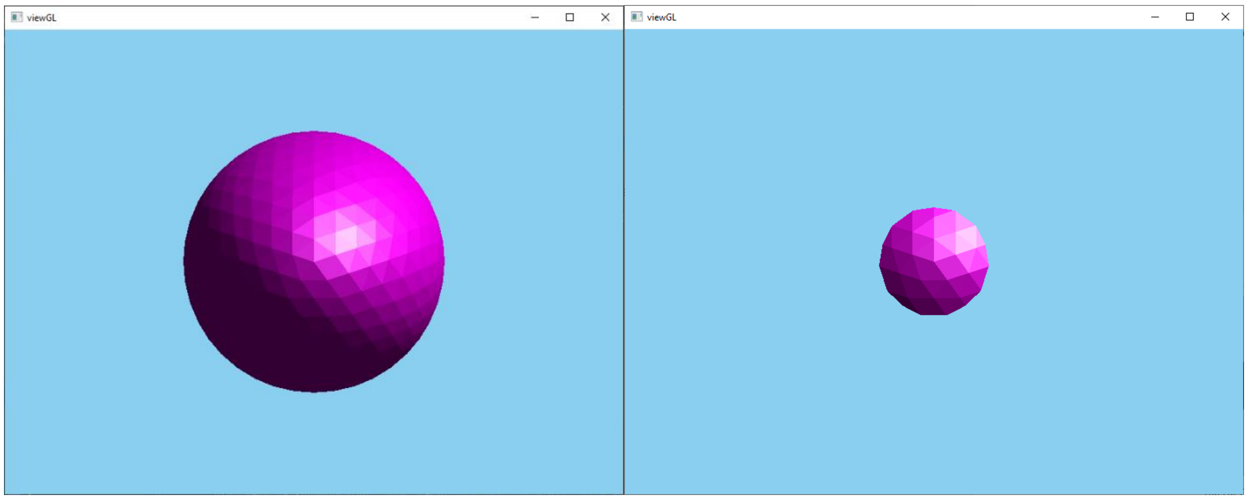
Near distance can be varied to change the minimum distance away a vertex needs to be from the camera to be rendered.



0.2 Near Distance vs 9 Near Distance

Far Distance

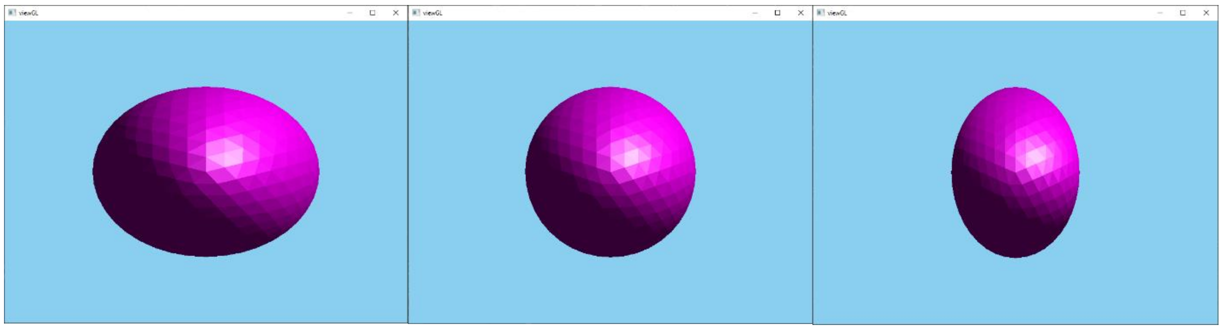
Far distance can be varied to change the maximum distance away a vertex can be from the camera to still be rendered.



100 Far Distance vs 8.65 Far Distance

Aspect Ratio

Aspect ratio can be varied to change how the object is stretched (note, the following images were rendered in a 4:3 window, so 4:3 aspect ratio is when the sphere is not distorted).



1:1 vs 4:3 vs 16:9

Part 2: Z Buffer Display

Three different z's in the depth buffer can be visualized by normalizing them to a [0,1] range and displaying that value as a grayscale color (white is farther, black is closer).

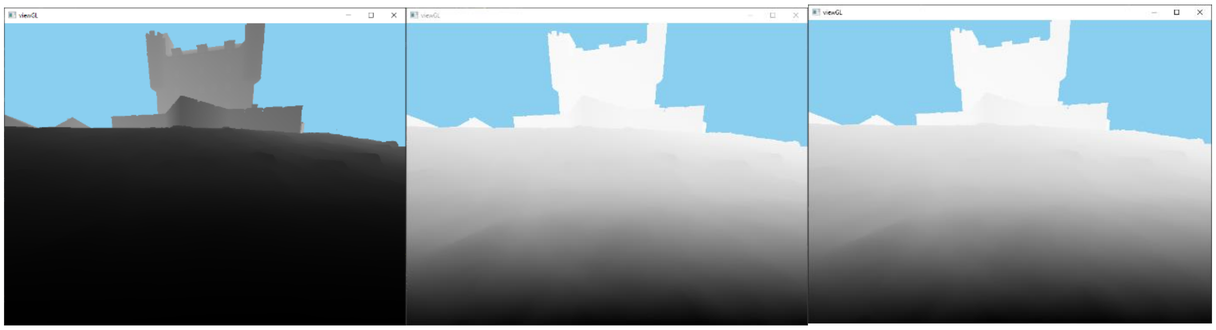
ZMode visualizes the Z coordinate of the vertex in the view space (before the projection matrix has been applied). This coordinate is in the range [n, f] where n is the near distance and f is the far distance, and is normalized into the [0, 1] range via z-n/f-n.

ZTildeMode visualizes the Z coordinate of the vertex in the clip space (after the projection matrix but before the perspective divide). This coordinate is in the range [-w, w] where w is the distance from the camera in view space. It is normalized into the [0, 1] range via z+w/2w (note, this normalization essentially divides by the w coordinate, making it equivalent to ZPrimeMode).

ZPrimeMode visualizes the Z' coordinate of the vertex in the screen space (after the perspective divide). This coordinate is in the range [-1, 1] so it is normalized into the [0, 1] range with z'+1/2.

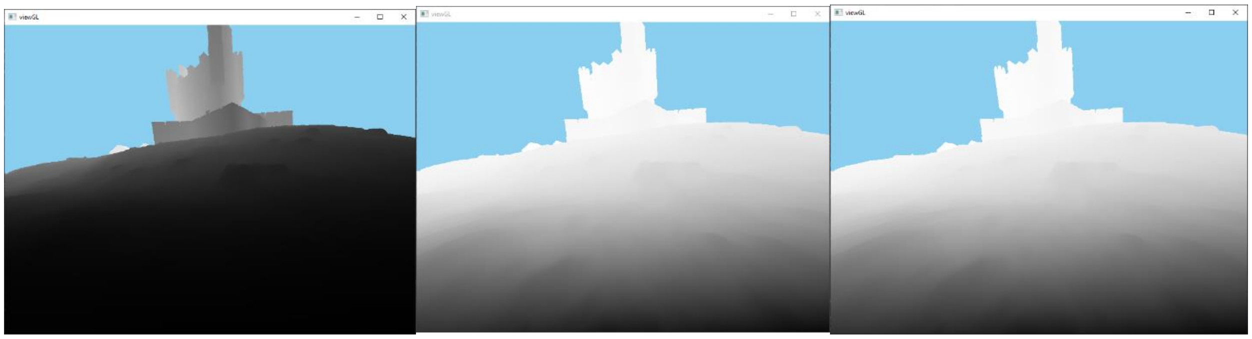
The Z Buffer is visualized in the following pictures are two different angles (to verify consistency).

Angle 1:



ZMode vs ZTildeMode vs ZPrimeMode

Angle 2:

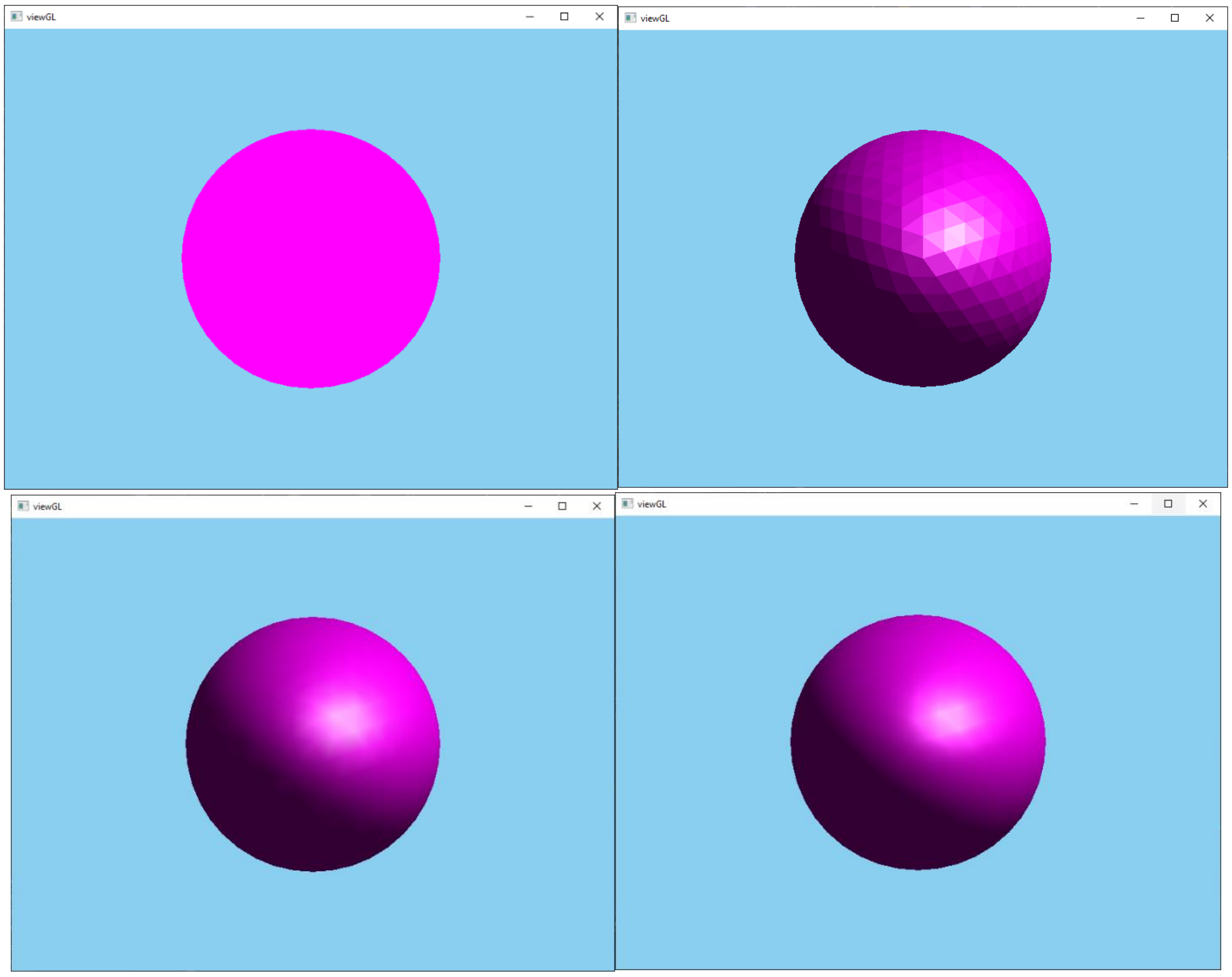


ZMode vs ZTildeMode vs ZPrimeMode

The main difference between the ZMode and ZPrimeMode (equivalent to ZTildeMode) coordinates is that ZMode shows a linear change in Z with distance. However, ZPrimeMode has a logarithmic change, where there is a drastic color change at the objects closest/farthest in the scene (the only black and gray is on the ground right next to the camera in ZPrimeMode while ZMode has a linear gradient of black into white).

Part 3/Bonus: Flat and Gouraud and Phong Shading

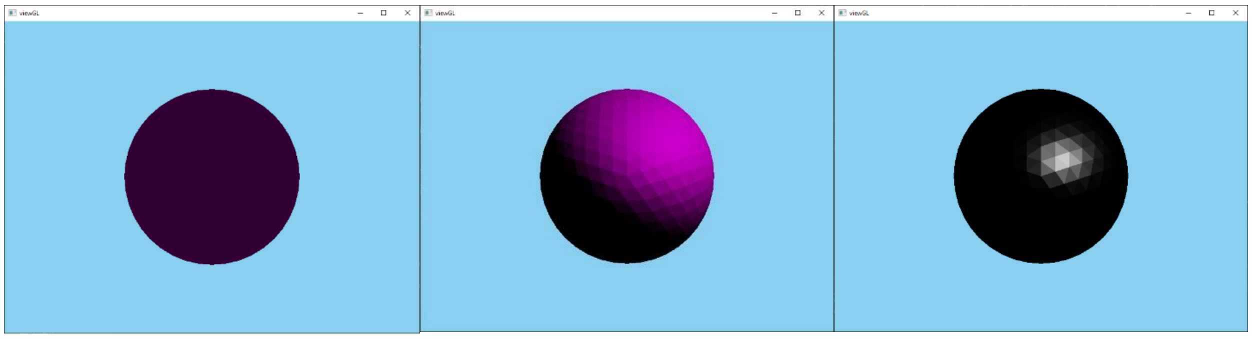
Ambient, diffuse, and specular shading can be applied as either Flat, Gouroud, or Phong in the shader from the directional light specified in the main file settings:



No Shading vs Flat Shading vs Gouraud vs Phong

Flat Shading

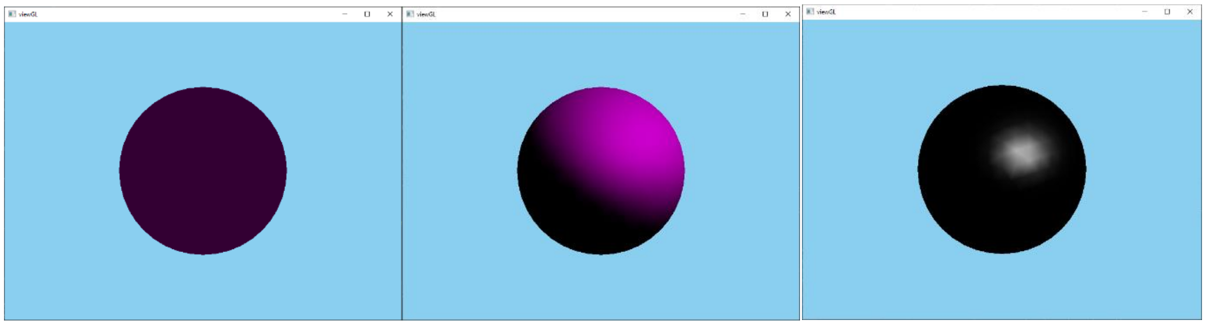
The normal used in flat shading is the normal of the triangle the vertex is a part of. This results in every triangle having a single color, creating a more geometric type of look:



Ambient Only vs Diffuse Only vs Specular Only

Gouraud Shading

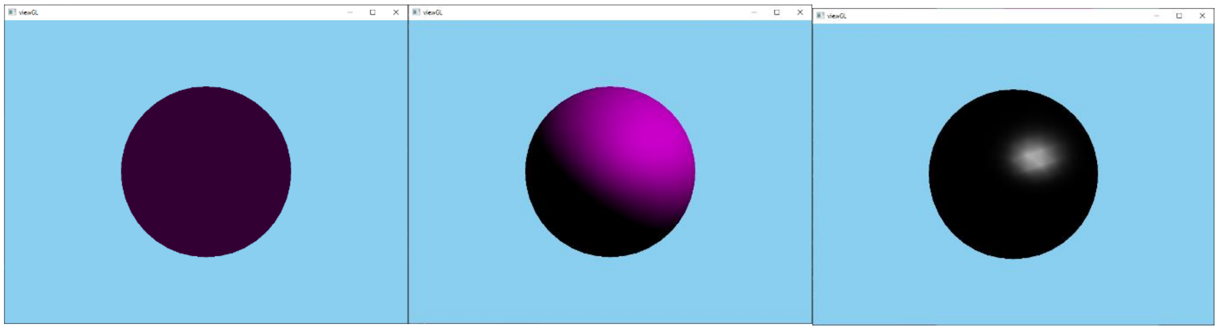
The normal used in Gouraud shading is the normal of the vertex (average of all triangle normal the vertex is a part of). This results in each vertex having its own shaded color which is interpolated to create the fragment color:



Ambient Only vs Diffuse Only vs Specular Only

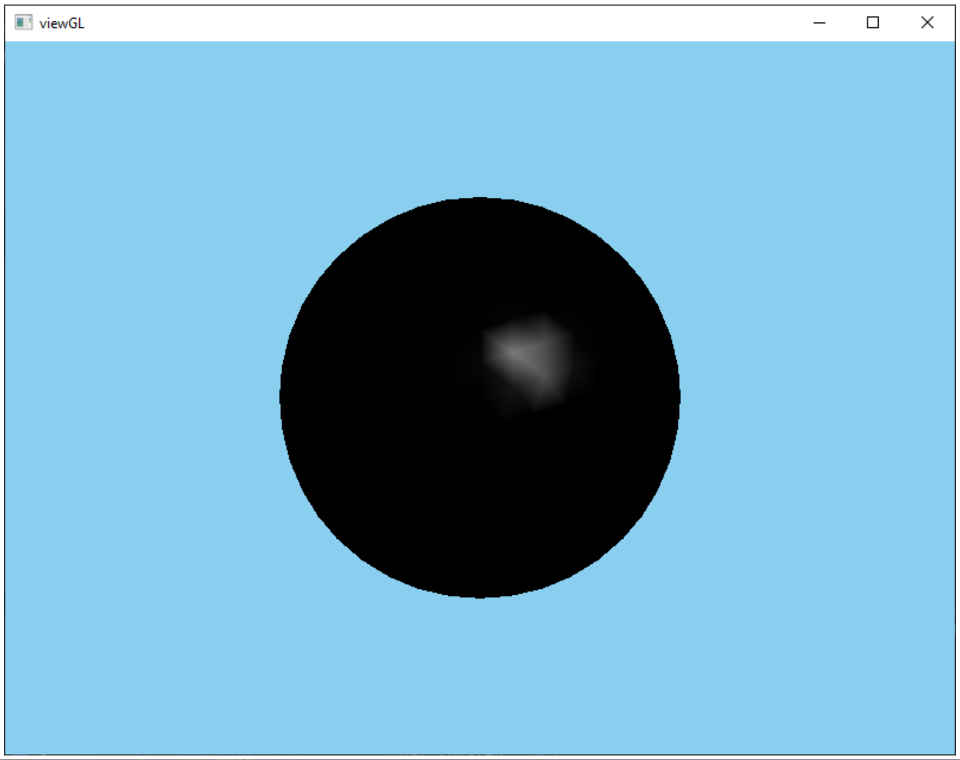
Phong Shading

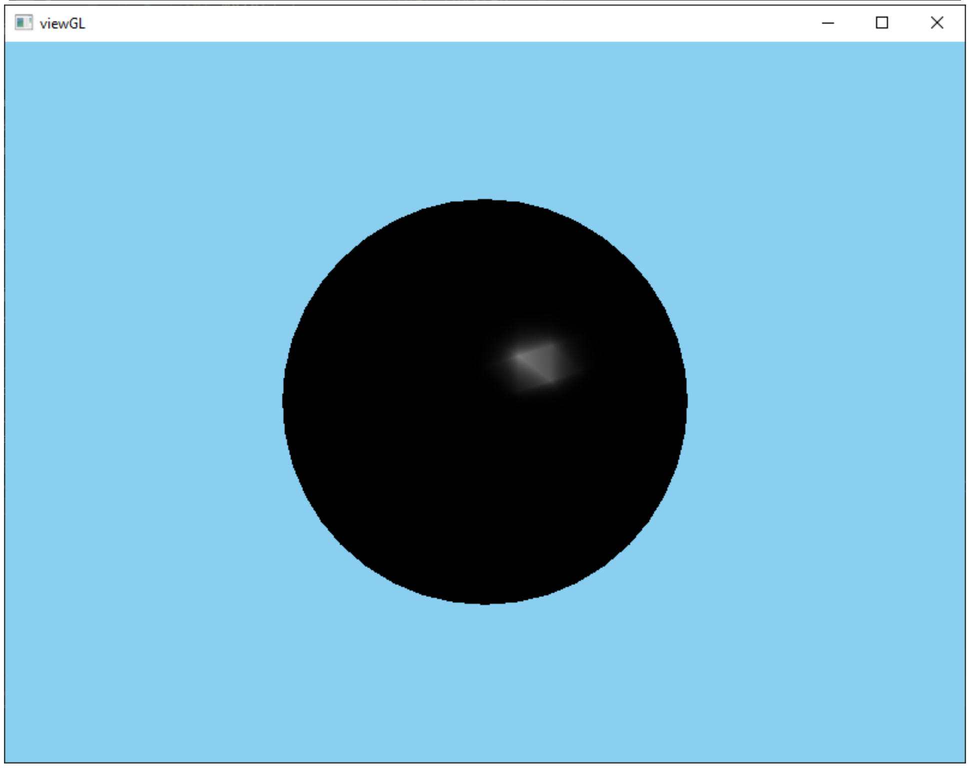
The normal used in Phong shading is the interpolated normal of all the vertex normal producing similar shading to Gouraud, except that the specular highlight is more well defined and circular:



Ambient Only vs Diffuse Only vs Specular Only

Here is a larger picture of Gouraud (top) and Phong (bottom) Specular highlights at a higher phong exponent to better show the difference between the two:





Video Demo

A video demo of the functionality detailed in this report can be found at: https://www.youtube.com/watch?v=KA77mCskGpU

Sources

This project used the GLFW, GLEW, and GLM Libraries. Other than this, no outside libraries were used, and no code was copied or repurposed from any sources, except for repurposing the initial main file from the professor. The following sources were referenced for technical information:

- Main file has portions of code from the initial main file provided by the professor.
- https://learnopengl.com/Getting-started/Coordinate-Systems#:~:text=The%20view%20space%20is%20what,camera%20space%20or%20eye%20space%20) was referenced for pipeline info
- https://www.scratchapixel.com/lessons/3d-basic-rendering/perspective-and-orthographic-projection-matrix/building-basic-perspective-projection-matrix.html#:~:text=This%20matrix%20now%20not%20only,are%20correctly%20interpreted%20for%20rendering and https://learnopengl.com/Advanced-OpenGL/Depth-testing were referenced for learning about the projection transform and Z buffer.
- Sphere obj file was found at https://thangs.com/designer/GeorgeDebarr/3d-model/sphere.obj-217158
- Castle obj and mtl file was found at https://sketchfab.com/3d-models/braemar-castle-aberdeenshire-scotland-7ebeeff19ead4b3997d1b3262cf56ea6