**Assignment 3b**

**CS4610**

**Michael Rallo – msr5zb – 12358133**

**3/23/2017**

**Assignment 3b**

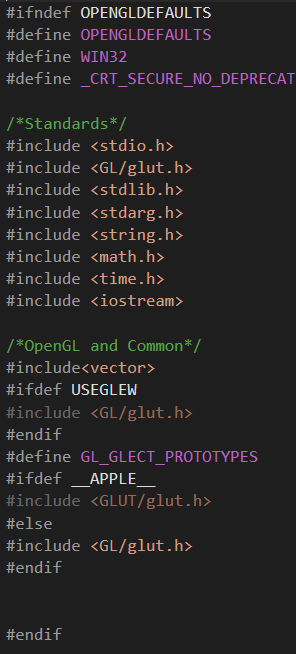
The *purpose* of this assignment was to familiarize yourself with the OpenGL command of the texture mapping.

Objectives:

1. Read a obj file [cube.obj](http://faculty.missouri.edu/duanye/course/cs4610-spring-2017/assignment/cube.obj) and an image file (.jpg) as a texture (here are two sample images: [checker-board](http://faculty.missouri.edu/duanye/course/cs4610-spring-2017/assignment/checker_256x256.jpg), [mandrill](http://faculty.missouri.edu/duanye/course/cs4610-spring-2017/assignment/mandrill_256x256.jpg), you can also use any other images you like) , and render the object with the texture mapped.
2. The texture should be modulated by the shading computation.

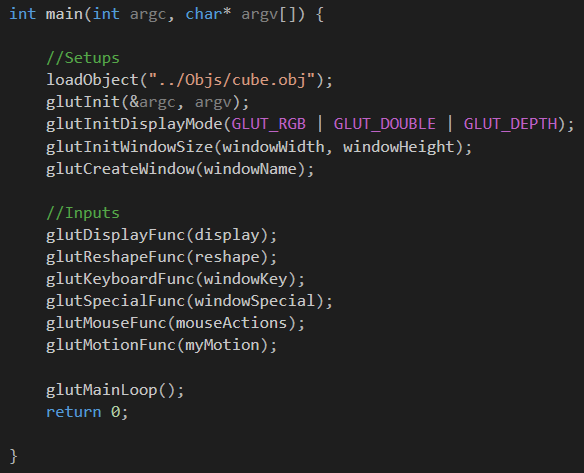
Approach: By adding the stb\_image.h library, I was able to load in textures. Once an image file was loaded in, values for the texture mapping/binding were implemented. The most difficult part of this assignment was coming up with an algorithm to calculate the glTexCoord2f() parameter values. I developed an algorithm that looks at planes a face’s vertices are on and then computes appropriate coordinates for the texture to object map.

**The Header File (OpenGLDefaults.h)**

First and foremost, I have decided to include a header file to be used for this assignment’s, as well as future assignments’, libraries. Note this file include OpenGl basic libraries, as well as printing for debugging and math for easy/complex calculations.

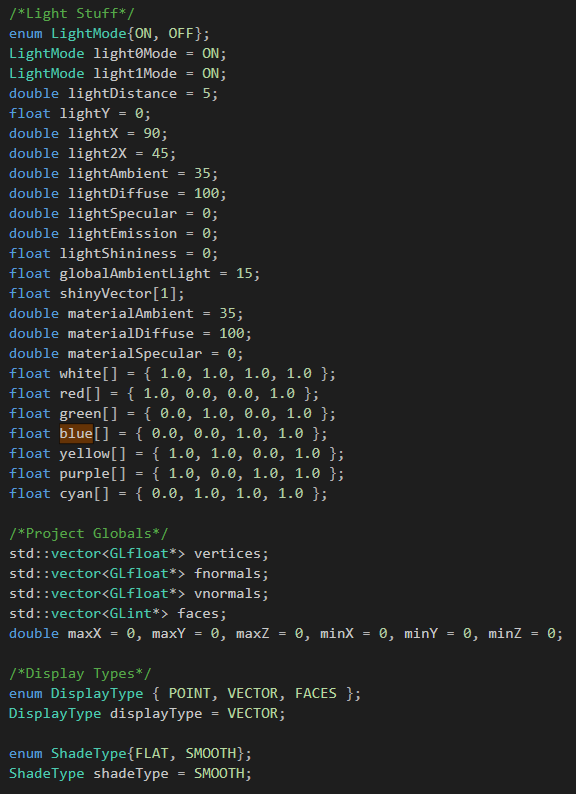
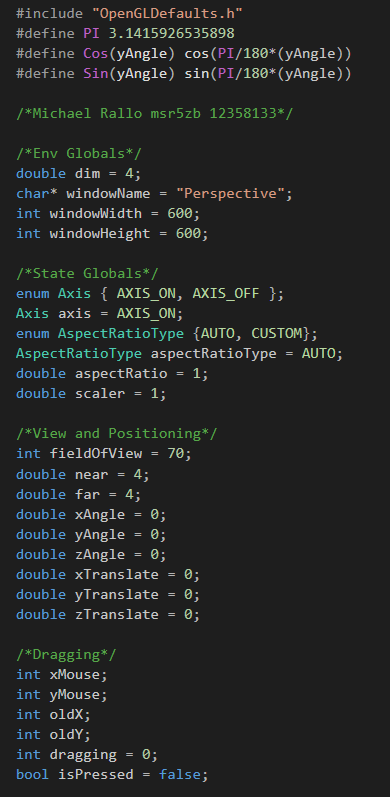
stb\_image.h was also added in to handle image loading and manipulation.

**Main**



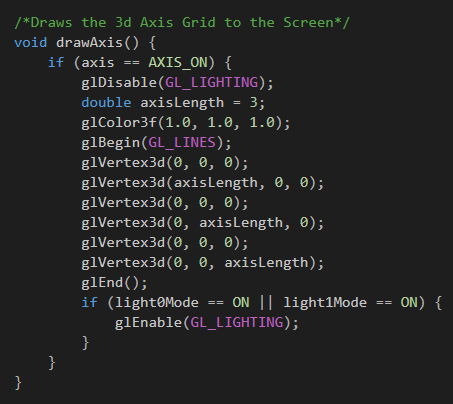
Our main function is similar to that from Assignment 3a. Note by default we load in the cube object file.

**Global Variables**



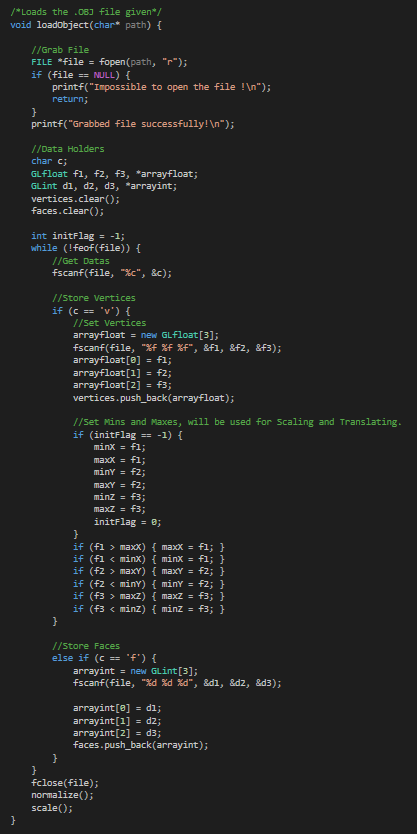
For this assignment, I will be using global variables in order to change various settings (named conventionally). Note the fieldOfView, near, and far to handle clipping and camera view. Lighting variables for the lights, environment, and objects were also created. Positions for lights will also be manipulated so we can see the full effects of lighting in our program.

**Drawing the XYZ Grid (Extras)**

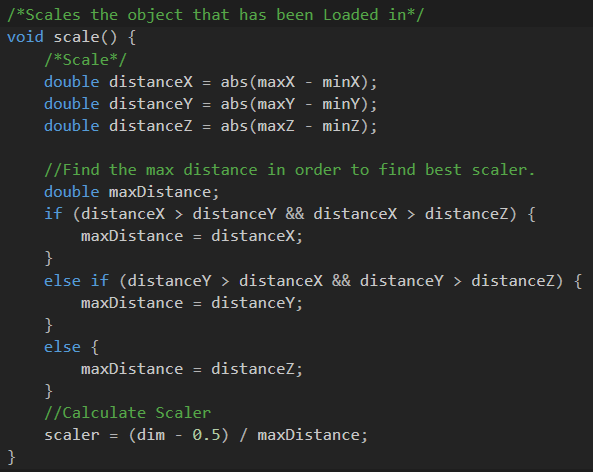


This is a simple function to draw a grid at the origin of our view in order for us to see the object more clearly. This can be toggled on and off with the “i” key. By default, it is on.

**Loading the File (loadObject)**

****

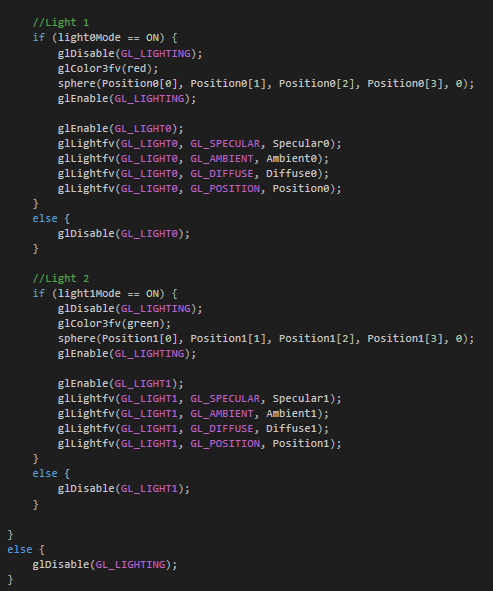
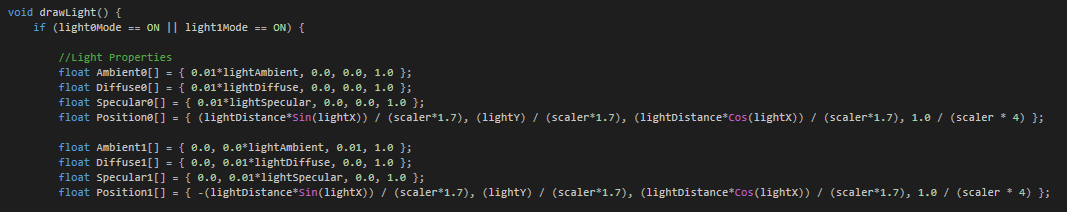
This function take the object the as a parameter and sets our global vertices and faces variables with the data the OBJ file contains. This function also sets the min/max values that will be later used for scaling/transitioning our object.

**Scaling (Scale)**

This scale method finds the greatest distance between the X, Y, and Z axis and uses that as a scaler for this Object. The reason we use the longest distance is so that we can scale everything equally whilst still being in our view.

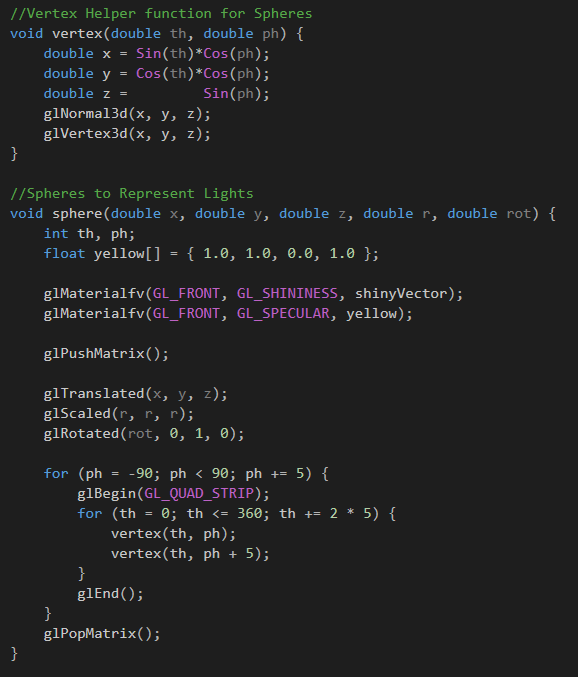
**LightSources**

This is our function to create our light sources. Note to adjustable RGBA values for each light. Also note how we use the scalar on the light source to keep a constant size throughout objects. Light Sources are toggle-able as well.



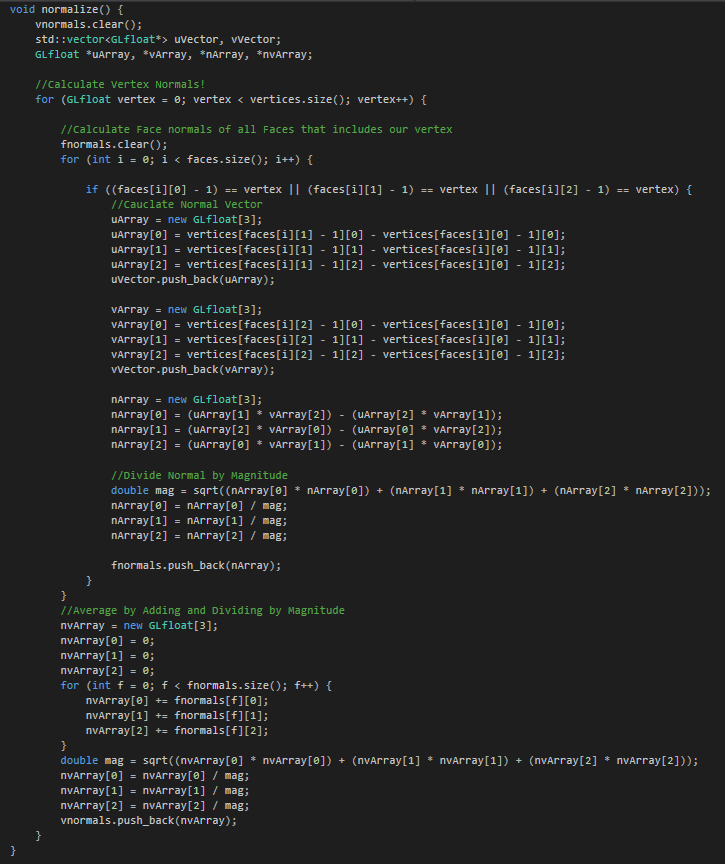
**LightSource Spheres**

**T**his function simply puts a sphere where the light source radiates from.



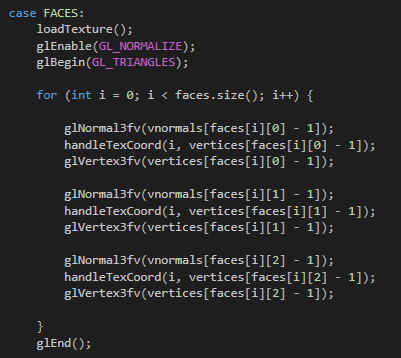
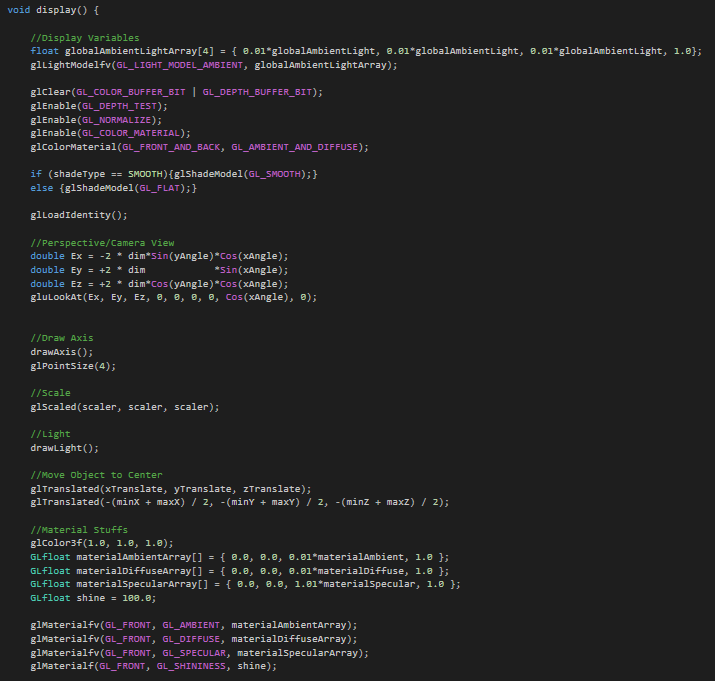
**Normal Vectors**

As described above, this is our ‘normalize’ function that calculates the normal vectors for all vertexes.



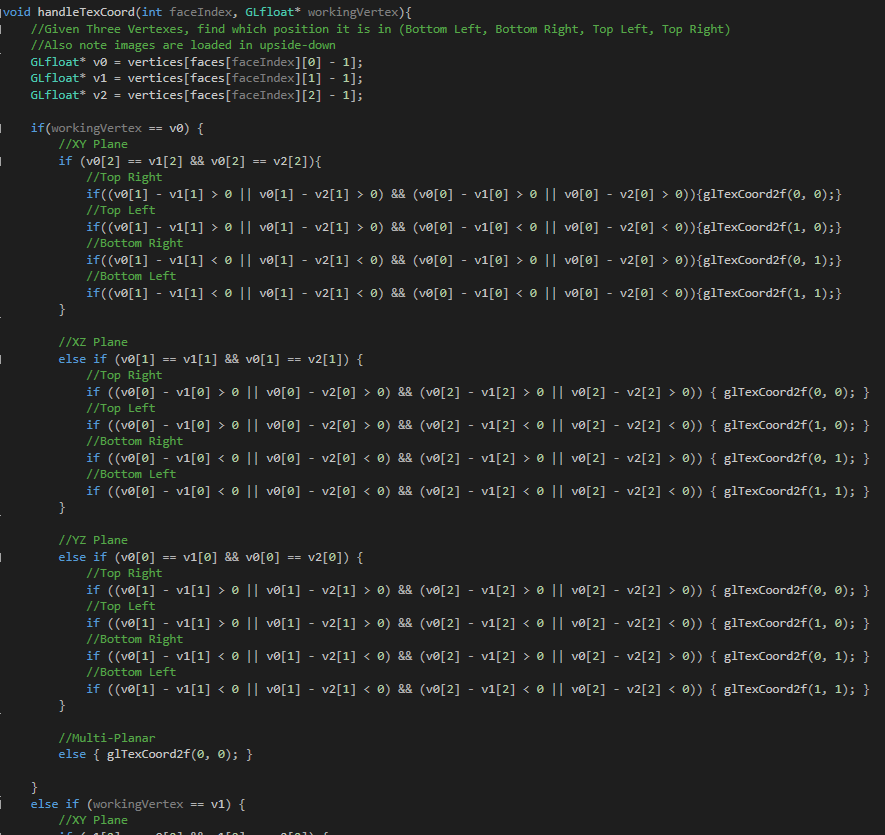
**Display**

Our display function has been updated to only handle perspective camera view. Important variables are also set here (depending on flags set). One of which is the Shadetype(GL\_SMOOTH/GL\_FLAT). Also notice the adjusted material values put in place for the objects to be created.



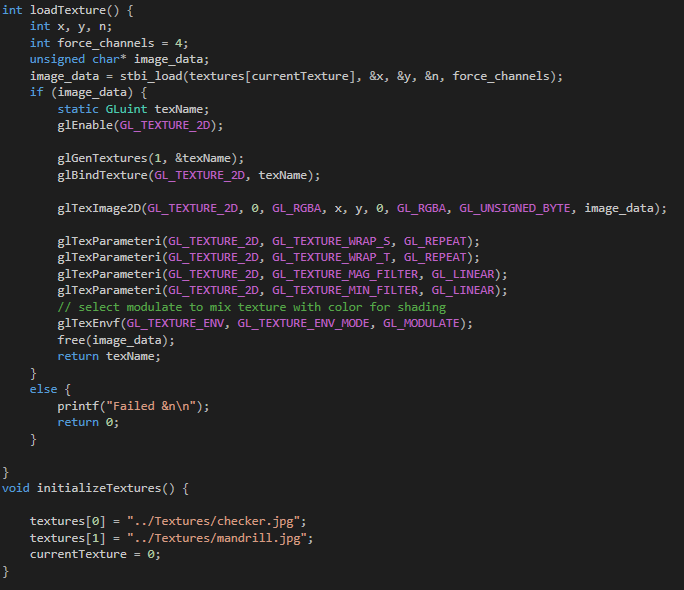
Note the handleTexCoord() call. This function is described on the next page. Also note the shading computation.

**HandleTexCoord**



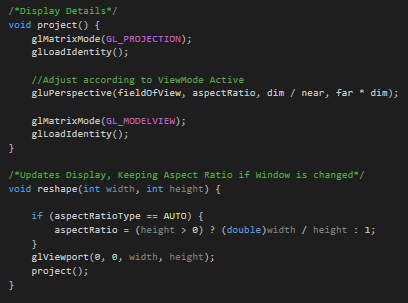
This function was used to calculate the appropriate glTexCoords() for a given face. It compares vertices via planes and then calculates a triangle that closest represents its intended texturing map. Note images in OpenGL are loaded in upside down – which was taken into account in the algorithm.

**Loading the Texture**



Using the stb\_image.h function stbi\_load() I was able to easily load the data of a desired image. I kept the different textures in an array so that user will be able to choose from multiple textures.

**Reshape**



Our reshape function is still very basic and has been redesigned to only handle projection/perspective mode rather than offering an orthographic option. Also note the Aspect Ratio Adjustability.

**Keyboard Input**

Key: Esc exits the program.

Keys: 1,2,3 loads in different Objects.

Keys: a,s,d changes display type.

Key: 4 changes textures of objects.

Key: i toggles the XYZ grid.

Keys: +,- Zooms In/Out (field of view).

Keys: b,B Scales Objects.

Keys: t,T Changes the Aspect Ratio.

Keys: g,G controls the clipping range for the near value.

Keys: f,F controls the clipping range for the far value.

Keys: l,L toggles lights 1 and 2.

Keys: <,> rotates the lights.

Keys: [,],{,} moves lights.

Keys: o,O adjusts the Global Ambience.

Keys: h,H adjusts lightAmbient.

Keys: j,J adjusts the lightDiffuse.

Keys: k,K adjusts the lightSpecular.

Keys: n,N adjusts the materialAmbient.

Keys: c,C adjusts the materialDiffuse.

Keys: v,V adjusts the materiaSpecular.

Key: q toggles the ShadeType.

Keys: x,X,y,Y,z,Z Translates objects.

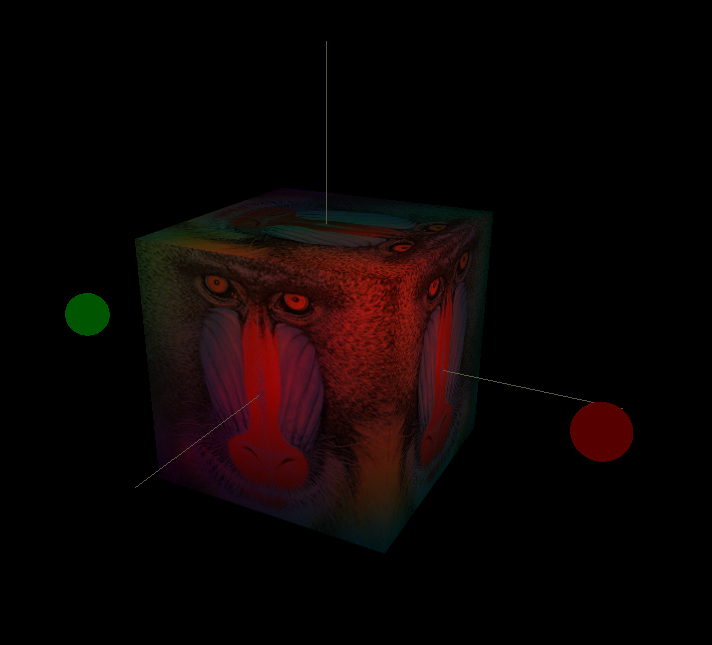
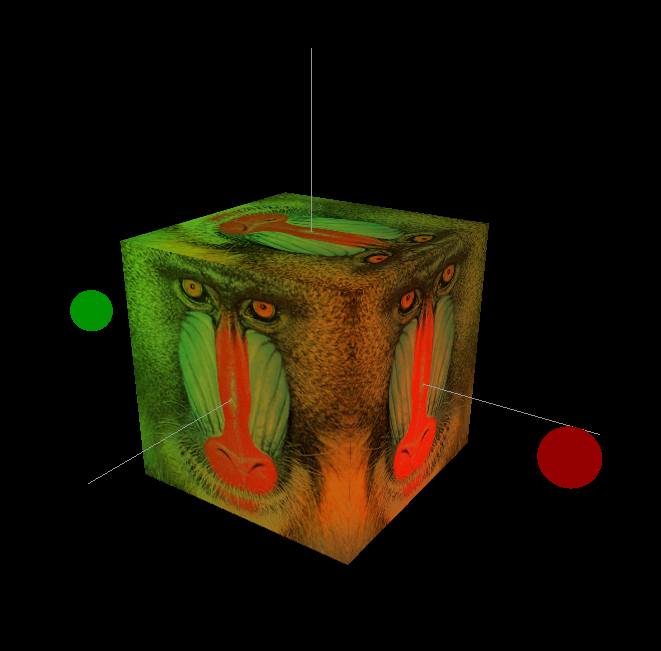
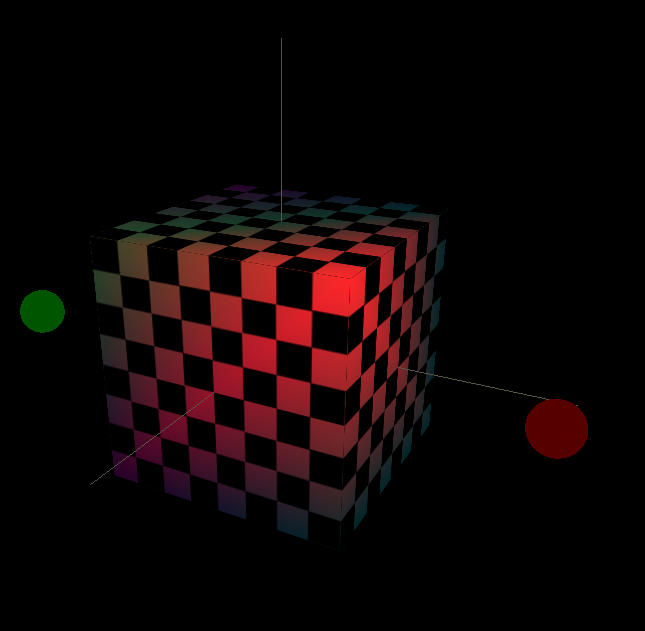
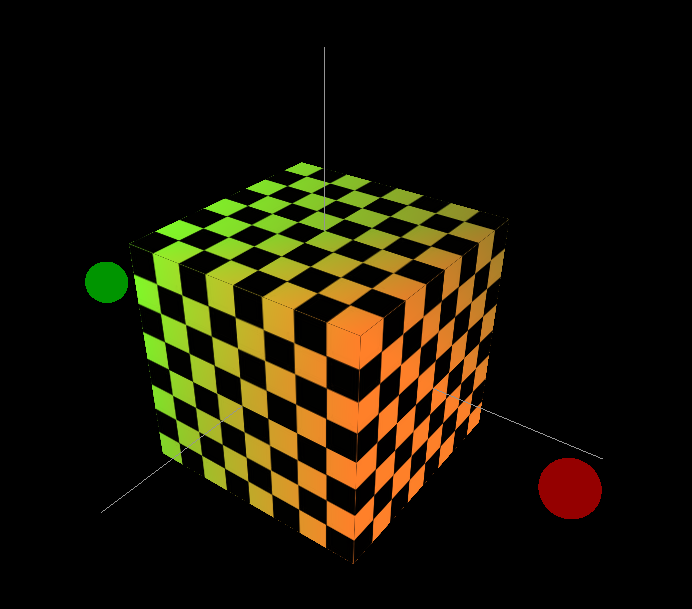
Key: r resets the object.

Arrow Keys adjusts/rotates object/view incrementally.

Click and Drag to Rotate Object, further you drag quicker it rotates.

**The Output**

The following are samples of outputs. Note how the Spheres represent the lightsources. Also note how the smoothing technique drastically affects how our objects looks. Adjusting the ambience, diffuse, and specular yields us very creative and interesting results as well! Textured objects still keep shading properties. Also, because textures are loaded into OpenGL upside-down, I flipped the glTexCoords so they display correctly.



Everything ran pretty smoothly, however coming up with the glTexCoords was pretty difficult.