1. Camera Rotation

How would you compute the matrix for a rotation of 90 degrees about the axis (1,1,1)?

In order to compute the matrix for a rotation of 90 degrees about the axis (1,1,1), you would would use Rodrigues' rotation formula (slide 59+) to compute different coordinates. This works out to this matrix: (slide 62)

$$\begin{bmatrix} 0 + u_x^2 & u_xu_y - u_z & u_xu_z + u_y \\ u_xu_y + u_z & u_y^2 & u_yu_z - u_x \\ u_xu_z - u_y & u_yu_z + u_x & u_z^2 \end{bmatrix}$$

where u for each component is the component of the normalized r (axis) vector, which in this case is (⅓, ⅓, ⅓). Thus,

$$\begin{bmatrix} 1/9 & -2/9 & 4/9 \\ 4/9 & 1/9 & -2/9 \\ -2/9 & 4/9 & 1/9 \end{bmatrix}$$

2. Shader Uniforms

Given a texture object of **m_tex1** and a shader object **m_shader**, please write pseudocode to set **m_tex1** to be set to a uniform variable declared as **uniform sampler2D texture1** contained in **m_shader**'s source code. Assume nothing about the current state!

Psuedocode:

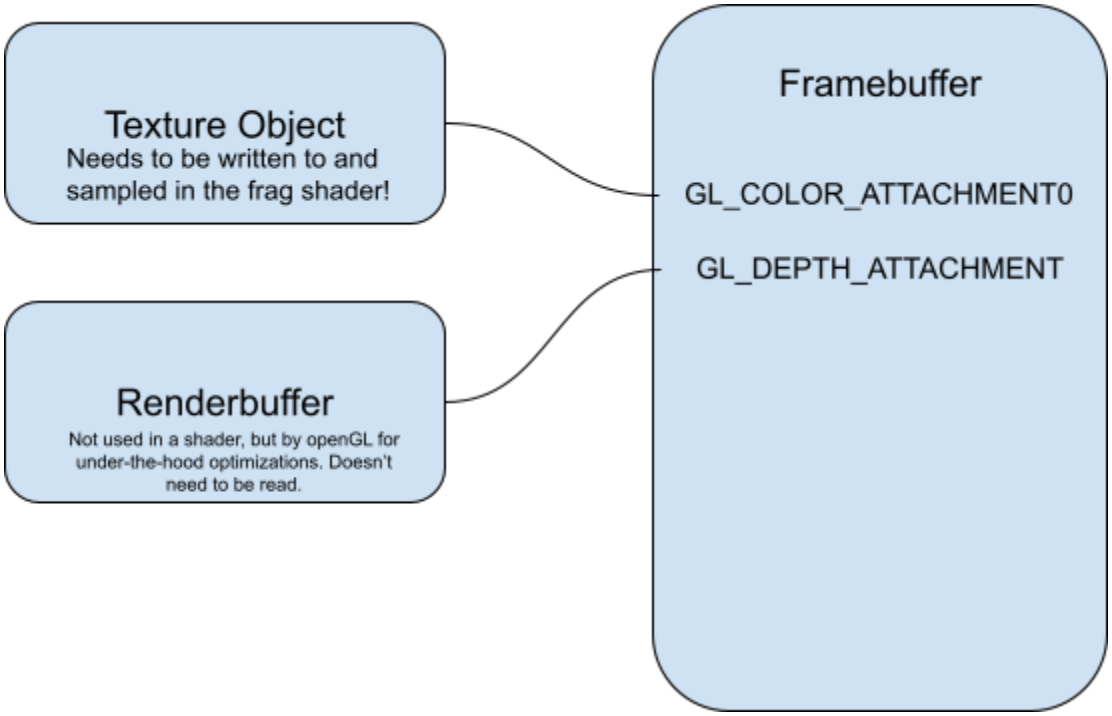
- Use the **m_shader** program
- Find location of "texture1" in the shader program
- Set 1i uniform in texture1 in the shader program to be **m_tex1**

Is this supposed to be more complex? In the case that you need to set up the texture, a lot more needs to be done (setting active textures, loading in images, combining renderbuffers, unbinding stuff) but if it's just loading in a uniform, it's pretty simple.

3. Textures, Renderbuffers, Framebuffers

a. Concept

For a framebuffer with color and depth attachments, please draw a schematic/diagram of the framebuffer indicating which attachments are textures and which are renderbuffers and why. Assume the usage of the framebuffer will be to draw a scene with a grayscale filter applied.



b. Functional Understanding

i. glTexImage2D

The difference between **format** and **internal format** in the glTexImage2D call is that internal format is the pixel formatting of the texture image that is loaded in, and format is the format of the image as accessed by the sampler2D uniform in the frag shader.

ii. glTexParameteri

When specified in glTexParameteri, GL_NEAREST will return the value of the nearest neighbor at a certain coordinate when doing mipmapping. GL_LINEAR, however, will provide a linear interpolation of the nearby coordinates to that pixel. Thus, GL_LINEAR will have a smoother scaling process than GL_NEAREST.

4. Realtime vs. Ray

a. Recursive Reflections

We are not implementing recursive recursions in this assignment because reflections are far, far more complex without raytracing. With raytracing, we are shooting out a path that can be bounced and followed along. With this algorithm, however, it is complex matrices that are flattening the vertices to the screen (and things get warped with the perspective matrices anyways). Plus, the vertex and fragment shaders run on one vertex at a time, so it would be complex to create relationships between vertices to create reflections.

b. Differences

Advantages of offline rendering other than reflections include shadows and no perspective warping toward the edges of the screen. Additionally, even though it's hard to see with granular realtime rendering, non-triangle-mesh primitives can be used and far fewer jaggies and artifacts show up, which allow for more precise coloring/lighting.

Advantages to realtime rendering include, primarily, speed and quicker changes. Additionally, mesh rendering, procedural texturing, and other normal-based operations are easier and quicker.