Introduction

In this assignment, we will add material properties to the wall. The material properties will determine the color of the object and the object's reflectivity in the light model. Thus, we will no longer use the `vColor` vertex attribute. You can continue to use

```
struct VertexData3D
{
    vec3 pos;
    vec3 color;
    vec3 norm;
};
```

in which case you must leave the color vectors in the data array, but you may (or may not) comment out or remove the statements

```
glBindVertexArray(VAO[WallArray]);
glVertexAttribPointer(vColor, 3, GL_FLOAT, GL_FALSE, ...
glEnableVertexAttribArray(vColor);
```

In any case, in the vertex shader, you may delete or ignore the `vColor` value and in the fragment shader either delete it or not use it in the calculations.

An alternative is to delete all references to `vColor` throughout the program, the `Wall` class, and the shaders and redefine the `struct` to be

```
struct VertexData3D
{
    vec3 pos;
    vec3 norm;
}
```

I recommend that you not do that at this time.

The User Interface

The user interface is the same as in Assignment 11. The lighting effects should be identical to Assignment 11, provided you give the wall the same color that it had in that assignment.

The Material Properties

This material properties will include the ambient reflection, the diffuse reflection, the specular reflection, and the shininess. Each is a 3-dimensional vector except for the shininess, which is a `float`. The ambient and diffuse properties are normally the same as the inherent color

of the object. The specular reflection is normally white, in order to make the shiny spot white, although on some physical objects, the color of the shiny spot is the same as the color of the object.

There will no longer be a shininess property of the light. We want that property to be different for different objects, so it becomes part of the object. These variables should be initialized and used in the usual way, but they should be defined and implemented within the `Wall` class, as suggested by the `m_` prefix on each variable. The values of these variables should be global constants defined in `global.h`.

On the other hand, the shader locations of these values should be global because those locations will be used by all objects.

```
GLint mat_amb_loc;
GLint mat_diff_loc;
GLint mat_spec_loc;
GLint mat_shiny_loc;
```

The `setMaterial()` Function

Create a `setMaterial()` member function in the `Wall` class that will send to the fragment shader the values of the uniform variables `mat_amb` and `mat_diff` to the color of the object (defined as a constant in `globals.h`), and assign to `mat_spec` a gray level, depending on the shininess of the object, and `mat_shiny` whose value determines the size of the shiny spot. This function should then be invoked by the `draw()` member function.

The Vertex Shader

The vertex shader will be the same as in Assignment 11 unless you choose to eliminate `vColor` (your choice).

The Fragment Shader

In the fragment shader, you will need to add the uniform variables `mat_amb`, `mat_diff`, `mat_spec`, and `mat_shiny` (and remove `light_shiny`) and incorporate then into the formulas in place of `vColor` and `light_shiny`. Again, you may remove `color` or leave it there, your choice. Just don't use it.

Due Date

This assignment will not be collected, but should be completed by Wednesday, October 30. In the next assignment, we will build the canal boat.