Lab Assignment #3
Due Oct. 29, 2019

Per-fragment Phong Lighting

The goal of this lab is to implement the Phong lighting model in a fragment shader. Specifically, you will compute lighting in world space using the local viewer assumption. Be sure to pass the relevant matrices to the vertex shader. Ignore the effects of nonuniform scaling when transforming normal vectors. In other words, you do not need to compute a normal matrix

Use the inverse square attenuation model in this assignment. This is equivalent to the quadratic attenuation model with a=0, b=0, c=1.

- Point light source. Place a point light source in the scene above the mesh.
  - Create an imgui slider which lets you move the light nearer to / further from the mesh. Give your slider enough range that the effect of attenuation is clearly visible. **[10 pts]**
  - Let the light ambient, diffuse and specular colors all be shades of grey. Create sliders to vary the intensity of L_a, L_d, L_s between 0.0 and 1.0. **[10 pts]**

- Material color.
  - Create imgui ColorEdit3 widgets to control the material ambient, diffuse and specular colors, k_a, k_d, k_s. **[10 pts]**
  - Create a slider to set the shininess (specular exponent) to values between 0.0 and 100.0 **[10 pts]**
  - Add a checkbox to control texture application. When unchecked your program should use the ambient and diffuse colors set by the widgets. When checked k_a, k_d should be set to the texture color. **[10 pts]**

- Correctly compute the ambient term of the Phong lighting model **[10 pts]**
  - This term is not attenuated.

- Correctly compute the diffuse term of the Phong lighting model **[10 pts]**
  - Transform normal vectors into world space.
  - For clarity, in glsl name your world space normal vector `nw` and your world space light vector `lw`.
  - Be sure to normalize `nw` and `lw`.
  - Remember to correctly clamp to avoid negative reflectances.

- View vector (local viewer). **[10 pts]**
  - In the fragment shader, compute a variable named `vw` (the world-space vector that points from the surface to the camera) by subtracting world space fragment position from eye position, then normalizing.

- Reflection vector. **[10 pts]**
    - Use the glsl reflect function to compute the reflection vector in the fragment shader
    - Remember to account for the fact that reflect() assumes the light vector points toward the surface: `vec3 rw = reflect(-lw, nw);`

- Correctly compute the specular term using rw and vw. **[10 pts]**
    - Remember to correctly clamp to avoid negative reflectances.
    - Be sure to normalize `rw` and `vw`.

Hints:

- Use your sliders to help debug. For example, you can debug the diffuse term by "zeroing-out" the contributions from the ambient and specular term by setting ambient and diffuse light intensities or material colors to zero. I will ask you to do this when grading so that I can see the 3 terms of the lighting equation independently, so be sure you understand how to do this.

- It may also help to debug intermediate variables by displaying them as colors. For example, use `fragcolor = vec4(nw, 1.0);` to visually inspect the world space normal vector.

- Be sure all points and vectors are in world-space before computing lighting

You will demo this program in class on the due date and submit your code to Blackboard. Be sure that you can demonstrate your program without recompiling code.