CS 450: Assignment 06

Programming Assignments (95%)

- Copy src/app/Assign05.cpp and name it **src/app/Assign06.cpp**
    - Similar to before, make sure the shaders are loaded from the **shaders/Assign06** folder (instead of shaders/Assign05)
- Make a copy of the shaders/Assign05 folder and name it **shaders/Assign06**
- Modify **CMakeLists.txt** by adding the following lines to the end of the file:

```
add_executable(Assign06 ${GENERAL_SOURCES} "./src/app/Assign06.cpp")
target_link_libraries(Assign06 ${ALL_LIBRARIES})
install(TARGETS Assign06 RUNTIME DESTINATION bin/Assign06)
install(DIRECTORY shaders/Assign06 DESTINATION bin/Assign06/shaders)
```

- Make sure the sample configures, compiles, and runs as-is

shaders/Assign06/Basic.vs

- **BEFORE the main() function:**
    - Add a new input: **layout(location = 2) in vec3 normal;**
    - Add a new uniform matrix for the normal transform: **uniform mat3 normMat;**
        - **NOTE: mat3!!!!!**
    - Add an output variable for the position after the model and view transformations: **out vec4 interPos**
    - Add an output variable for the normal after the normal transformation: **out vec3 interNormal**
- **IN the main() function:**
    - Calculate the position AFTER model and view transformations only and set **interPos** to that value
    - Calculate the normal AFTER normal transformation and set **interNormal** to that value

shaders/Assign06/Basic.fs

- **BEFORE the main() function:**
    - Add an input variable for the position AFTER the model and view transformations: **in vec4 interPos**
    - Add an input variable for the normal AFTER the normal transformation: **in vec3 interNormal**

    - Add a struct to hold a point light:
        - **struct PointLight {**
          vec4 pos;
          vec4 color;
          };
    - Add a uniform variable for a PointLight: **uniform PointLight light;**

- **IN the main() function:**
    - Normalize interNormal → N
    - Calculate the light vector L as the NORMALIZED direction vector FROM interPos TO lightPos (you will have to convert to vec3 at some point)
    - Calculate the diffuse coefficient (float) as the max of 0 and the dot product of N and L
    - Multiply the diffuse coefficient by the vertexColor and light.color and convert to vec3 → vec3 diffColor
    - Assume a shininess of 10.0 for now
    - Calculate the specular coefficient, incorporating the multiplication by the diffuse coefficient
        - The pow() function exists in GLSL
    - Compute the specular color
    - Set out_color to be the diffuse plus the specular color

src/include/MeshData.hpp

- **Ensure the Vertex struct** has the following field: **glm::vec3 normal**

src/lib/MeshGLData.cpp

- **Ensure that the following is implemented in the function createMeshGL:**
    - Enable vertex attribute array 2 (this will be for the normals): **glEnableVertexAttribArray(2);**
    - Use **glVertexAttribPointer()** to allow OpenGL to correct read in the normal data: **glVertexAttribPointer(2, 3, GL_FLOAT, GL_FALSE, sizeof(Vertex), (void*)offsetof(Vertex, normal));**

src/app/Assign06.cpp

- **Add struct PointLight** with the following fields:
    - glm::vec4 pos
    - glm::vec4 color

- **Create a global PointLight object *light***

- **Modify the function extractMeshData** to also get the normals per vertex:
    - When getting the position and color for each Vertex, also grab the normal from mesh->mNormals[i]
      REMEMBER: these are aiVertex3D objects, so you will have to grab the x, y, z elements to store in a glm::vec3
    - **Change the default color of each vertex to (1,1,0,1) (yellow)**
        - If your background color causes that to blend in, change your background color to something else.

- **Modify renderScene by doing the following:**
    - **Add two more parameters**:
        - GLint normMatLoc
          glm::mat4 viewMat
    - **Add the following AFTER tmpModel is calculated but BEFORE the calls to drawMesh():**
        - **Calculate the normal matrix (glm::mat3)** as:
            - The transpose…
            - …of the inverse…
            - …of the glm::mat3…
            - …of viewMat times tmpModel (in the order)
        - **Use glUniformMatrix3fv to pass in the normal matrix**
            - **NOTE: Matrix3fv!!!!!!!!!**
    - **Modify renderScene's recursive call** to pass in the two new parameters: normMatLoc and viewMat

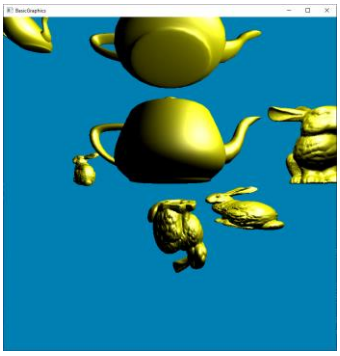
- **Add keys to your GLFW key callback function:**
    - If the action is either GLFW_PRESS or GLFW_REPEAT, add checks for the following keys:
        - GLFW_KEY_1
            - Set light.color to (1,1,1,1) (white)
        - GLFW_KEY_2
            - Set light.color to (1,0,0,1) (red)
        - GLFW_KEY_3
            - Set light.color to (0,1,0,1) (green)
        - GLFW_KEY_4
            - Set light.color to (0,0,1,1) (blue)

- **In the main function:**
    - **AFTER the creation of the shader program but BEFORE the rendering loop:**
        - Set the light position to (0.5, 0.5, 0.5, 1) and light color to (1,1,1,1)
        - Get the uniform locations of "light.pos" and "light.color"
        - Get the uniform location for "normMat"

- **INSIDE the drawing loop, AFTER the call to glUseProgram() and the calculation of the view matrix, but BEFORE the call to renderScene():**
  - Calculate the **position of the light in eye/view space**
    - Note that light.pos is initially in world space
  - **Use glUniform4fv() to pass in the VIEW-space position of the light**
    - **NOTE: 4fv!!!!!!**
    - **ALSO NOTE: NO "Matrix", just glUniform4fv!**
  - **Use glUniform4fv() to pass in the color of the light**
    - **NOTE: 4fv!!!!!!**
    - **ALSO NOTE: NO "Matrix", just glUniform4fv!**
  - **Modify the current call to renderScene** to pass in the two new parameters (the normMatLoc and the viewMat)

Screenshot (5%)

For this part of the assignment, take a screenshot of the application window when it first loads **bunnyteatime.glb**:



Grading

Your OVERALL assignment grade is weighted as follows:

- 95% - Programming
- 5% - Screenshot