CS 450: Assignment 07

Programming Assignments (95%)

- Copy src/app/Assign05.cpp and name it **src/app/Assign07.cpp**
  - Similar to before, make sure the shaders are loaded from the **shaders/Assign07** folder (instead of shaders/Assign06)
- Make a copy of the shaders/Assign06 folder and name it **shaders/Assign07**
- Modify **CMakeLists.txt** by adding the following lines to the end of the file:

```
add_executable(Assign07 ${GENERAL_SOURCES} "./src/app/Assign07.cpp")
target_link_libraries(Assign07 ${ALL_LIBRARIES})
install(TARGETS Assign07 RUNTIME DESTINATION bin/Assign07)
install(DIRECTORY shaders/Assign07 DESTINATION bin/Assign07/shaders)
```

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

shaders/Assign07/Basic.fs

- **BEFORE the main() function:**
  - Add two new uniform variables:
    - **uniform float metallic;**
    - **uniform float roughness;**
  - Add a constant float for PI = 3.14159265359
- Add a function: **vec3 getFresnelAtAngleZero(vec3 albedo, float metallic)**
  - This function calculates the external reflection R_F at incoming light angle 0.
  - Parameter metallic is assumed to be between 0 and 1.
    - If 0 → insulator (e.g., plastic), and albedo is diffuse color
    - If 1 → metal, and "albedo" becomes the specular color
  - Start with vec3 F0 at vec3(0.04)
    - Good default value for insulators
  - Use mix() function to interpolate between default F0 and albedo:
    F0 = mix(F0, albedo, metallic);
  - Return F0
- Add a function: **vec3 getFresnel(vec3 F0, vec3 L, vec3 H)**
  - This function returns the Fresnel reflectance given the light vector and half vector, assuming a starting value of F0 (i.e., $R_F(0)$).
  - Compute the max of 0 and the dot product of L and H → cosAngle
  - Use the **Schlick approximation** to calculate the Fresnel reflectance (see slide 38 of the PBR slides).
  - Return the computed value.

- Add a function: **float getNDF(vec3 H, vec3 N, float roughness)**
    - This function returns the Microgeometry Normal Distribution Function (NDF) value (i.e., how many microgeometry normals are aligned for reflection).
    - Use the GGX/Trowbirdge-Reitz NDF (see slide 45 of the PBR slides).
    - Return the computed value.
- Add a function: **float getSchlickGeo(vec3 B, vec3 N, float roughness)**
    - This is a helper function for getGF() (see slide 50 of the PBR slides).
    - Calculate k as $(\text{roughness} + 1)^2 / 8$
    - Calculate dot(N, B) / (dot(N, B)*(1 – k) + k)
    - Return computed value
- Add a function: **float getGF(vec3 L, vec3 V, vec3 N, float roughness)**
    - This function returns the Geometry Function value (i.e., how many microfacets are NOT shadowed or masked (see slide 50 of the PBR slides).
    - Compute GL = getSchlickGeo(L, N, roughness)
    - Compute GV = getSchlickGeo(V, N, roughness);
    - Return GL*GV
- **IN the main() function:**
    - Remove the existing out_color assignment.
    - Calculate the normalized view vector V (remember that interPos is in view space).
    - Calculate F0 using getFresnelAtAngleZero(vec3(vertexColor), metallic).
    - Calculate the normalized half-vector H.
    - Calculate Fresnel reflectance F with getFresnel(F0, L, H).
    - Set specular color kS to F.
    - Calculate the complete diffuse color as follows:
        - Set diffuse color kD to 1.0 – kS.
        - Multiply kD by (1.0 – metallic)
            - If metal → diffuse color does not exist.
        - Multiply by vec3(vertexColor).
        - Divide by PI.
    - Calculate the complete specular reflection (see slide 33 of the PBR slides) as follows:
        - Calculate NDF using getNDF(H, N, roughness).
        - Calculate G using getGF(L, V, N, roughness).
        - Multiply kS by NDF and G.
        - Divide kS by (4.0 * max(0, dot(N,L)) * max(0, dot(N,V))) + 0.0001.
    - Calculate final color as finalColor as (kD + kS)*vec3(light.color)*max(0, dot(N,L)).
    - Set out_color to vec4(finalColor, 1.0).

src/app/Assign07.cpp

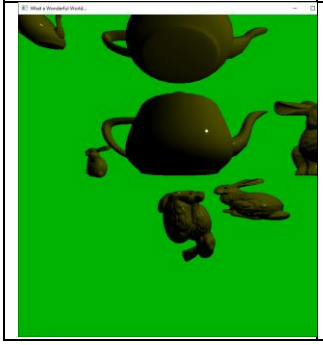
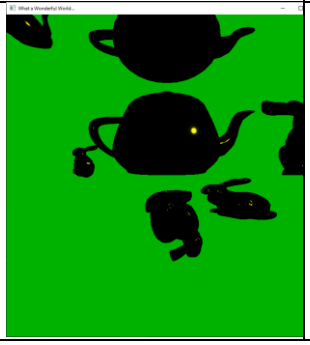
- **Add the following global variables:**
    - **float metallic = 0.0;**
    - **float roughness = 0.1;**

- **Add keys to your GLFW key callback function:**
    - NOTE: You will clamp:
        - metallic to [0,1]
        - roughness to [0.1, 0.7]
    - If the action is either GLFW_PRESS or GLFW_REPEAT, add checks for the following keys:
        - GLFW_KEY_V
            - Subtract 0.1 from metallic.
            - Make sure metallic does NOT drop below zero!
        - GLFW_KEY_B
            - Add 0.1 to metallic.
            - Make sure metallic does NOT exceed 1.0!
        - GLFW_KEY_N
            - Subtract 0.1 from **roughness**.
            - Make sure **roughness** does NOT drop **below 0.1**!
        - GLFW_KEY_M
            - Add 0.1 to **roughness**.
            - Make sure **roughness** does NOT **exceed 0.7**!

- **In the main function:**
    - **AFTER the creation of the shader program but BEFORE the rendering loop:**
        - Get the uniform locations for "roughness" and "metallic".
    - **INSIDE the drawing loop, AFTER the call to glUseProgram():**
        - **Use glUniform1f() to pass in the current metallic value**
        - **Use glUniform1f() to pass in the current roughness value**

Screenshot (5%)

For this part of the assignment, **upload FOUR screenshots** of the application window when it first loads **bunnyteatime.glb**:

Roughness = 0.1 ***Metallic = 0***	***Roughness = 0.1*** ***Metallic = 1***	***Roughness = 0.7*** ***Metallic = 0***	***Roughness = 0.7*** ***Metallic = 1***
			

Grading

Your OVERALL assignment grade is weighted as follows:

- 95% - Programming
- 5% - Screenshot