Lab 05: Standard Shading

Task 1: Visualize a triangle facing towards the +z direction.

Task 1.1: Define the triangle's vertex positions.

Task 1.2: Define the triangle's vertex normals.

Task 1.3: Construct the triangle as a Drawable.

Task 1.4: Construct the object (Suzanne) as a Drawable.

Task 1.5: Implement the drawing of either the triangle or the object.

Task 2: Model light and material properties and assign ambient intensity to the fragment color.

Task 2.1: Model light; specular (Ls), diffuse (Ld) and ambient (La) color.

Task 2.2: Model material properties; specular (Ks), diffuse (Kd), ambient (Ka) color and specular exponent (Ns). Use the following values, representing gold material.

Gold:

- specular: 0.628281, 0.555802, 0.366065, 1.0
- diffuse: 0.75164, 0.60648, 0.22648, 1.0
- ambient: 0.24725, 0.1995, 0.0745, 1.0
- shininess: 51.2

Task 2.3: Model ambient intensity (Ia)

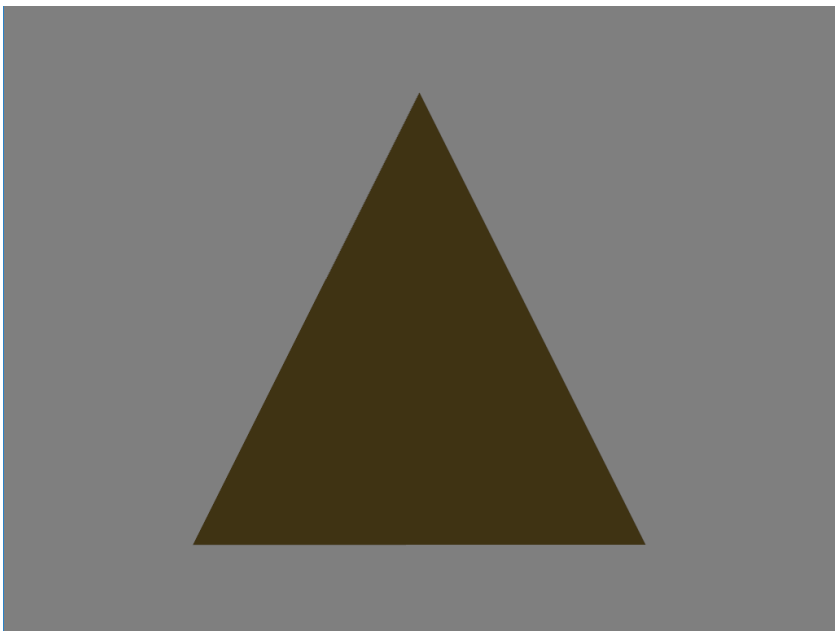


Figure 1. Ambient intensity for gold material.

Task 3: Model diffuse intensity.

Task 3.1: Transform the position of the vertex from model space to camera space. Propagate the result to the fragment shader.

Task 3.2: Transform the normal of the vertex from model space to camera space. Propagate the result to the fragment shader. Attention: vertex normal is a direction vector!

Task 3.3: Transform the position of the light from world space to camera space. Propagate the result to the fragment shader.

Task 3.4: Calculate the normalized vertex normal (N) in camera space.

Task 3.5: Calculate the normalized light direction (L) in camera space.

Task 3.6: Compute cos(θ), where θ is the angle between the normal (N) and the light direction (L). Clamp the value above 0, to account for the cases where the light is behind the surface (use clamp(float, min, max)).

- light is at the vertical of the triangle -> 1
- light is perpendicular to the triangle -> 0
- light is behind the triangle -> 0

Task 3.7: Calculate the diffuse intensity and the new fragment color.

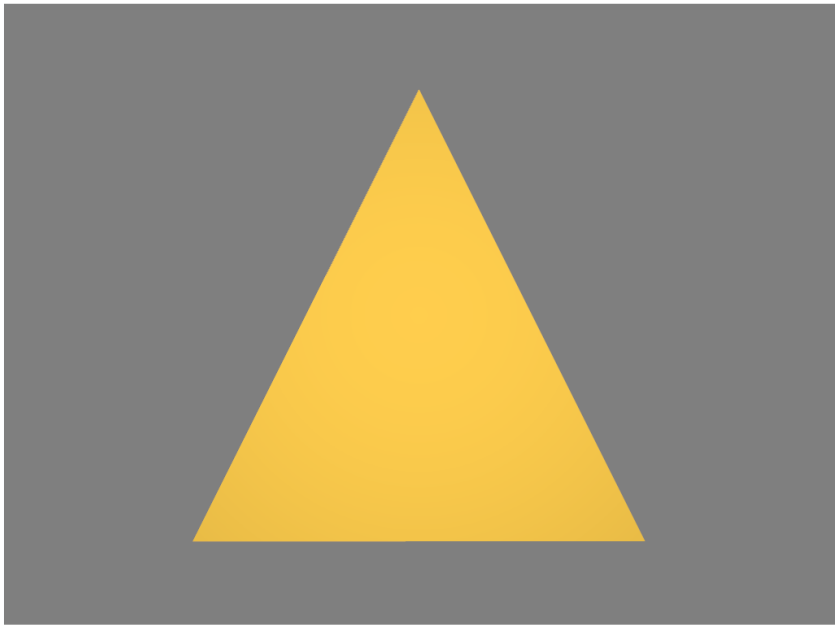


Figure 2. Ambient + Diffuse intensity for gold material.

Task 4: Model specular intensity

Task 4.1: Compute the direction of the light's reflection (R) in camera space.

Task 4.2: Compute surface-to-viewer direction ("Eye vector" – E) in camera space.

Task 4.3: Compute cos(α); where α is the angle between the eye vector (E) and the reflection vector (R). Clamp the value above 0, so the specular component will fade as α increases.

- looking into the reflection -> 1
- looking elsewhere -> <1

Task 4.4: Calculate the specular factor: (cos(α)) ^ Ns.

Task 4.5: Calculate the specular intensity and the new fragment color.

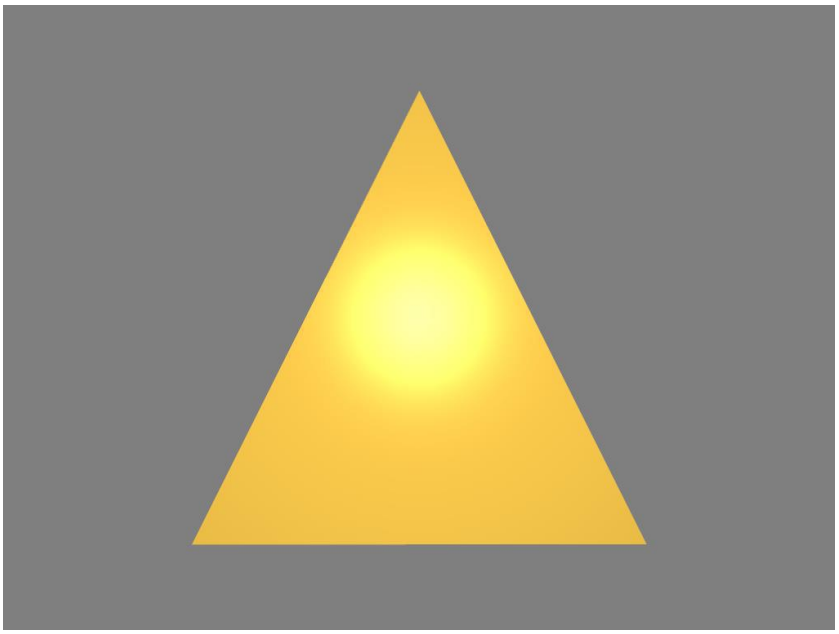


Figure 3. Ambient + Diffuse + Specular intensity for gold material.

Task 5: Model the light attenuation effect (add a light_power parameter). Change the light position to verify the correct behavior.

Task 6: Use the specular and diffuse textures to assign material properties to the Suzanne model.

Task 6.1: Draw the object instead of the triangle.

Task 6.2: Load diffuse and specular texture maps.

```
diffuseTexture = loadSOIL("suzanne_diffuse.bmp");
specularTexture = loadSOIL("suzanne_specular.bmp");
```

Task 6.3: Get a pointer to the texture samplers (diffuseColorSampler, specularColorSampler).

```
diffuseColorSampler = glGetUniformLocation(shaderProgram, "diffuseColorSampler");
specularColorSampler = glGetUniformLocation(shaderProgram, "specularColorSampler");
```

Task 6.4: Bind textures and transmit the diffuse and specular maps to the GPU.

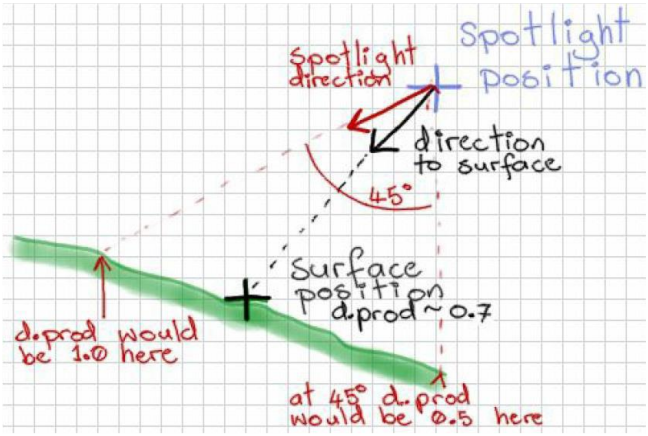
```
glActiveTexture(GL_TEXTURE0);
glBindTexture(GL_TEXTURE_2D, diffuseTexture);
glUniform1i(diffuceColorSampler, 0);

glActiveTexture(GL_TEXTURE1);
glBindTexture(GL_TEXTURE_2D, specularTexture);
glUniform1i(specularColorSampler, 1);
```

Task 6.5: Assign material properties according to the texture maps.

```
Ks = texture(specularColorSampler, vertex_UV);
Kd = ...
Ka = ...
Ns = 10;
```

Figure 4. Specular and diffuse materials from texture maps. Note the glass effect on the eyes.

Task 7: Create the spotlight effect



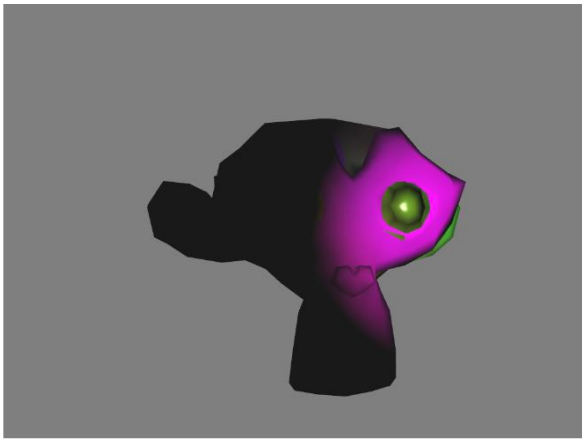


Figure 5. Spotlight model.

Homework

Homework 1: Modify the code for the spotlight in order to create a “ring light”. Use rough attenuation.

Homework 2: Implement Gouraud shading. Transfer all the shading calculations from “PhongShading.fragmentshader” over to “GouraudShading.vertexshader”. Calculate the vertex color (instead of the fragment color) and send the interpolated color data to the fragment shader.

Homework 3: Implement flat shading. Transfer all the shading calculations from “GouraudShading.vertexshader” over to “FlatShading.vertexshader”. Using the “flat” auxiliary qualifier, disable the interpolation of the color data that is being sent to the fragment shader.

Homework 4: Make model materials as uniform variables and display multiple instances of the model with different materials. You may use this table.

Homework 5: Make light properties as uniform variables and use the keyboard keys to adjust them (light position, color and power).

Homework 6: Choose one (or both!) of the following exercises:

> Homework 6A: Display three objects each being shaded by a different algorithm (flat, Gouraud, Phong) at the same time. You will have to switch between three shader programs and get the uniform locations for each one.

OR

> Homework 6B: Implement four more ring lights. Change their colors and positions in order to display the symbol of the Olympic Games, the Olympic Rings.

(Bonus) Homework 7: Load the object's vertices and UVs using *loadOBJWithTiny*. Ignore the normals returned by *loadOBJWithTiny* and try to compute them yourself. Create a Drawable using these newly computed normals. Compare with the true normals and try to improve the results. The lecture slides may come in handy!