Graphics programming
Exercise 8 - Lighting

Henrique Debarba

IT University of Copenhagen

Exercise 8

- Learning objectives
  - Understand the role of and differences between ambient, diffuse and specular reflections.
  - Use different light reflection models (Lambertian and Phong).
  - Implement and describe the difference between Gouraud and Phong shading.
  - Implement light attenuation.
  - Understand the separation between light and material attributes.

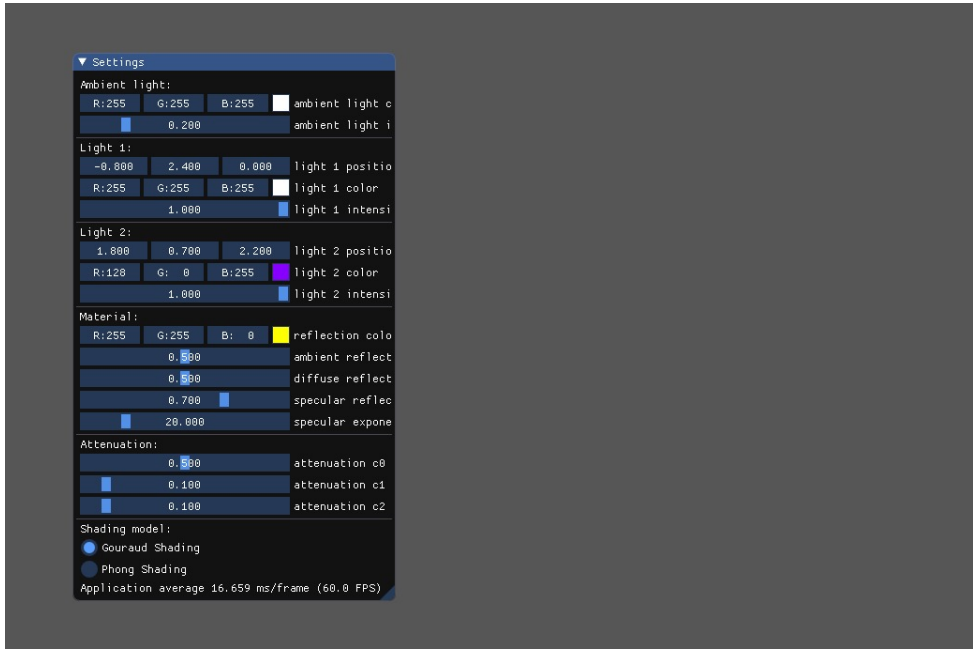
Exercise 8

- Additional resources
  - https://learnopengl.com/Lighting/Colors
  - https://learnopengl.com/Lighting/Basic-Lighting
  - https://learnopengl.com/Advanced-Lighting/Advanced-Lighting
  - https://en.wikipedia.org/wiki/Gouraud_shading
  - https://en.wikipedia.org/wiki/Phong_reflection_model

Exercise 8

- Initial implementation
  - there is a car in this image, if you move the Camera you might be able to see it.
  - This show how important lighting really is ☺

- Note:
  - We now use *Dear Imgui* (https://github.com/ocornut/imgui) for our graphical user interface (GUI). There you will be able to change the lighting parameters (color, intensity, shading model, etc…)
  - Press space to show/hide the GUI



Exercise 8

- For exercises **8.1 to 8.3**, we will work with the shaders:
  - **gouraud_shading.vert** and **gouraud_shading.frag**
- For exercises **8.4 to 8.6**, we will work with the shaders:
  - **phong_shading.vert** and **phong_shading.frag**
- Both implementations will use the same reflection model (the phong reflection model). The only difference is whether lighting values are computed:
  - for each vertex and interpolated for each fragment (AKA Gouraud shading),
  - or for each fragment (AKA Phong shading).
- You can use the GUI to swap between the two

8.1 Gouraud shading 1 (per vertex light)

- **Ambient reflection** is used as a crude approximation of **indirect lighting**. To implement you will need to:
- Send the following uniforms to the shaders
  - (ambientLightColor * ambientLightIntensity)
  - reflectionColor // the color of the object
  - ambientReflectance // how much ambient light it reflects
- Following the equations seen in class, compute the ambient reflection in the vertex shader and send the final color to the fragment shader.
- Can you see the car?

8.2 Gouraud shading 2 (per vertex light)

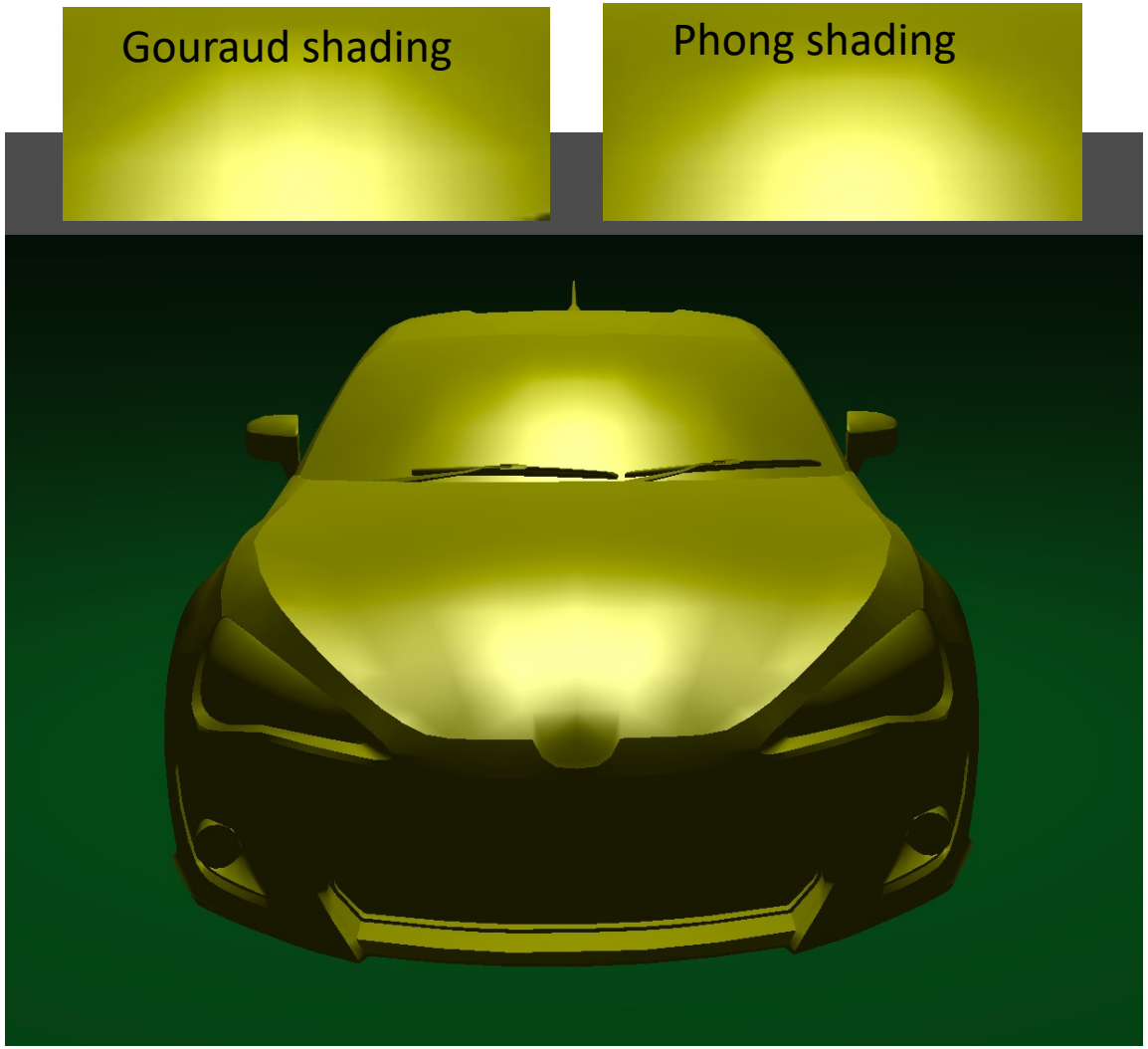
- **Diffuse reflection** is the light that is **scattered** and reflected in **every direction** as it hits the surface of the object. To implement it you will need to:
- Send the uniforms to the shaders
  - light1Position
  - light1Color * light1Intensity
  - diffuseReflectance // how much diffuse light it reflects
- Follow the equations in the class slides to compute the diffuse reflection in the vertex shader, then add it to the final color.
  - Make sure that all your positions and vectors are represented in the same space! We are computing lighting in world space in this exercise.

8.3 Gouraud shading 3 (per vertex light)

- **Specular reflection** is the light that gets reflected towards a **predominant direction**. It creates the spotlights that you can see in the image on the right.
- Send the uniforms to the shader
  - light1Position
  - light1Color * light1Intensity
  - specularReflectance // how much spacular light it reflects
  - specularExponent // how concentrated the spotlight is, the higher the value, the smoother is the surface of the material
- Following the equations seen in class, compute the diffuse reflection in the vertex shader and add it to the final color.
  - Make sure that all your positions and vectors are represented in the same space! We are computing lighting in world space in this exercise.
- Congratulations! You have completed a Gouraud shading implementation of the Phong reflection model!

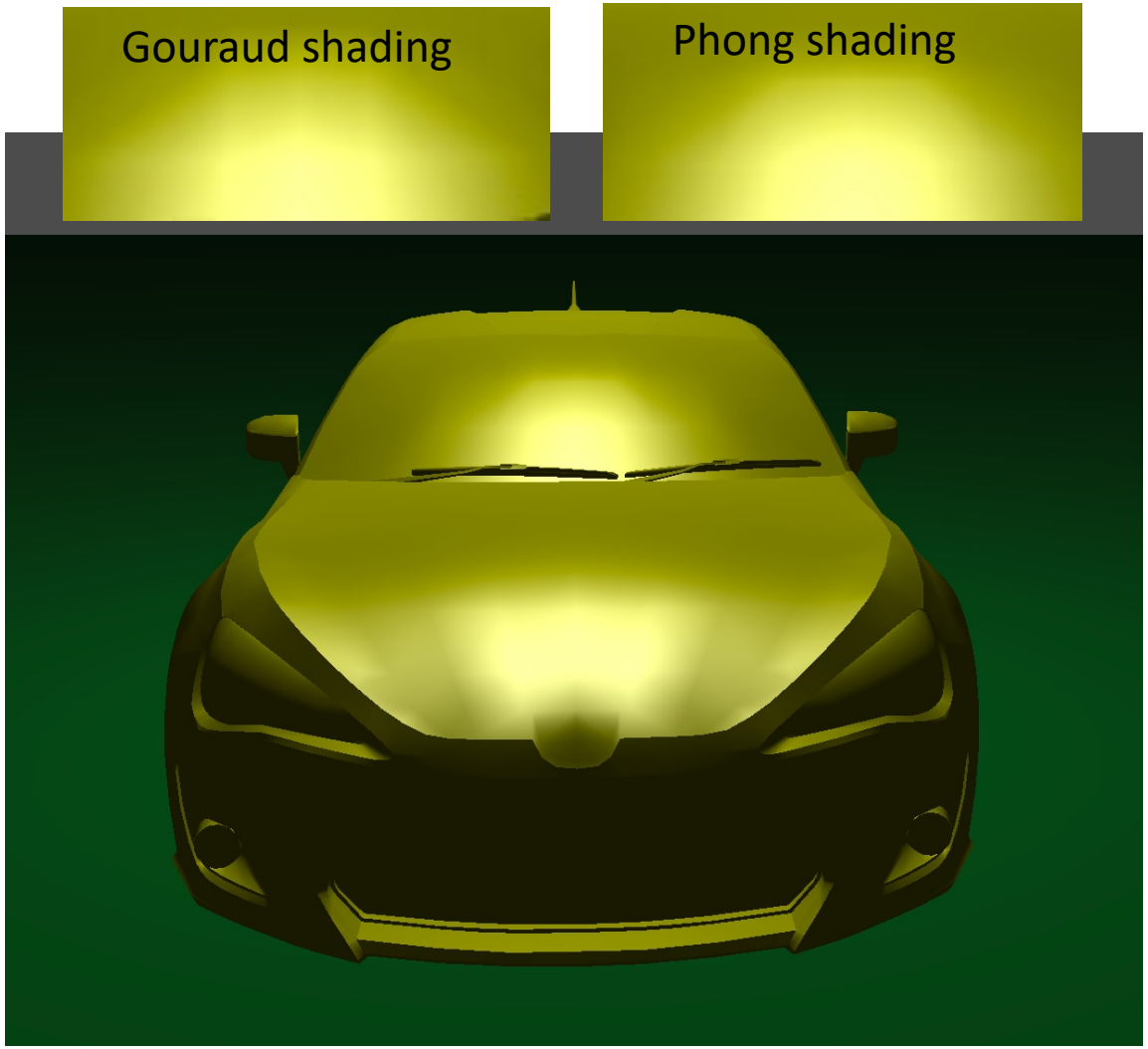
8.4 Phong shading (per pixel light)

- **Gouraud shading computes lighting per vertex**, and interpolate the final color of each pixel using the graphics pipeline. If the mesh has large triangles, this results in the triangular artifacts that you can see in the right.
- **Phong shading interpolates the normal of the surface** using the graphics pipeline, and computes the final lighting color only at the very end, in the **fragment shader**.
- To implement Phong shading:
  - Copy the lighting computation to fragment shader, from **gouraud_shading.vert** to **phong_shading.frag**)
  - You will need all the uniform variables from before on your fragment shader too.
  - You need to send the vertex **normal** and **position** from the vertex shader (phong_shading.vert) to the fragment shader using **out** and **in** variables. These will get interpolated in the rendering pipeline.
- Instead of interpolated shaded values, now you should get a smooth lighting effect.



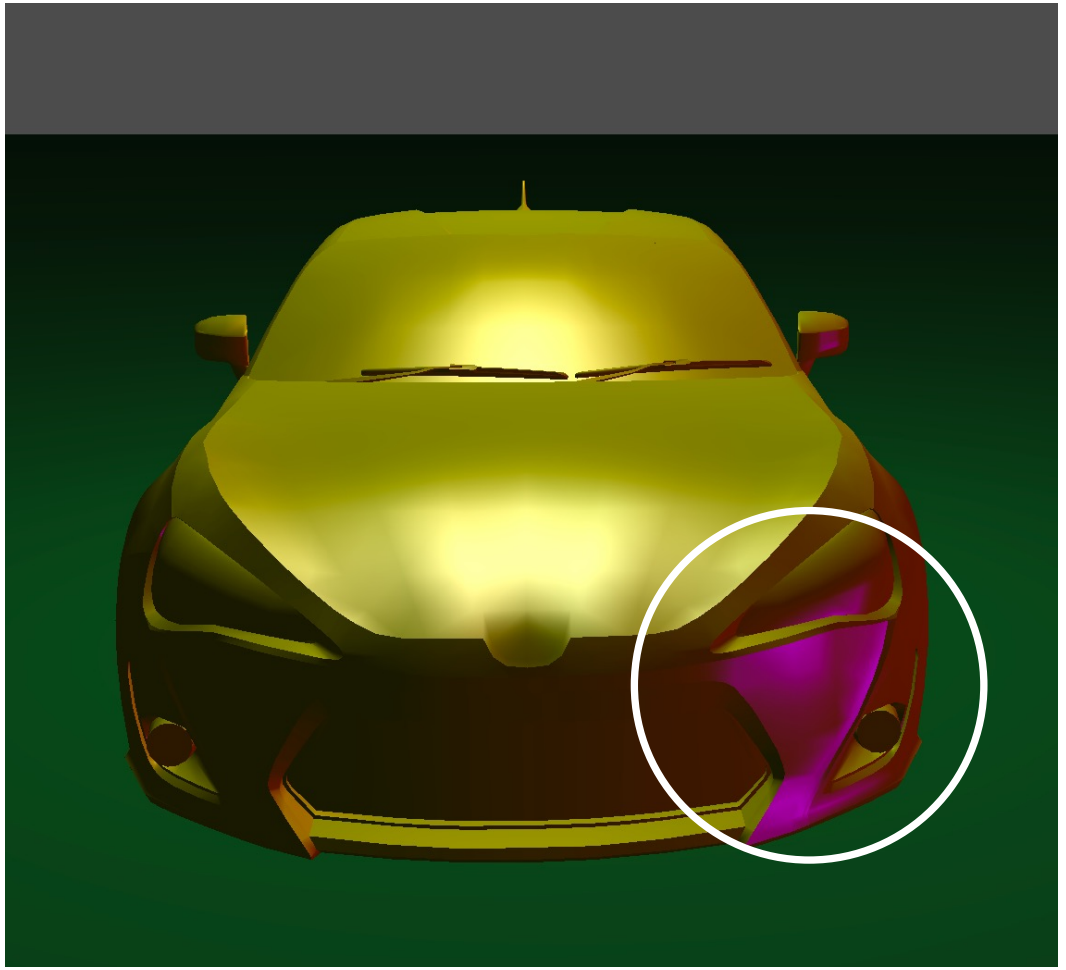
8.4 Phong shading (per pixel light)

- Congratulations! You have implemented the Phong reflection model!
- Notice how we can now see the reflection of the light on the floor. This was not possible with Gouraud shading, can you tell why this was the case?
  - Hint: the floor is made up by only two (very large) triangles, that make up a plane.

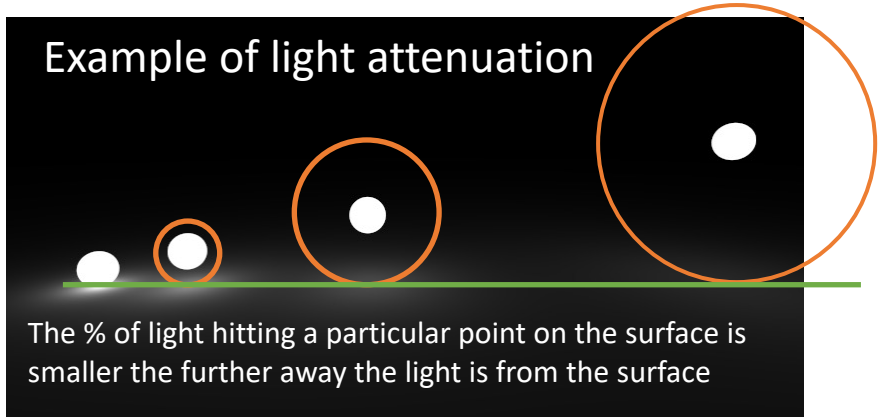


8.5 Two light sources

- Why stop here? Isn't the real world complicated and full of light sources?
- Add a second light source to your **phong_shading.frag** shader:
  - Send light 2 parameters to the shaders.
  - Compute diffuse and specular values for light 2.
  - Add light 2 diffuse and specular to final color.



8.6 Light attenuation



- Lighting loses intensity as the distance between the light emitter and the surface increases. That is because the further the light waves travel, the larger is the area of the spherical surface that it occupies. That means that the lighting area is increasing, but the energy is not (it remains constant)
- We can simulate this **attenuation** by taking the distance between the fragment and the light into account. To implement attenuation, send the uniforms to the fragment shader
  - attenuationC1
  - attenuationC2
  - attenuationC3
- Following the equations in the class slides, compute attenuation using the distance of the vertex to the light source and the 3 attenuation parameters above.