Graphics programming Exercise 8

Henrique Debarba

Knud Henriksen

IT University of Copenhagen

Important

- I updated to the submodule used in the exercises with the models we will use in the course: you should run the following command:
  - git submodule foreach git pull origin master

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 ☺)

8.1 Gouraud shading 1 (per vertex light)

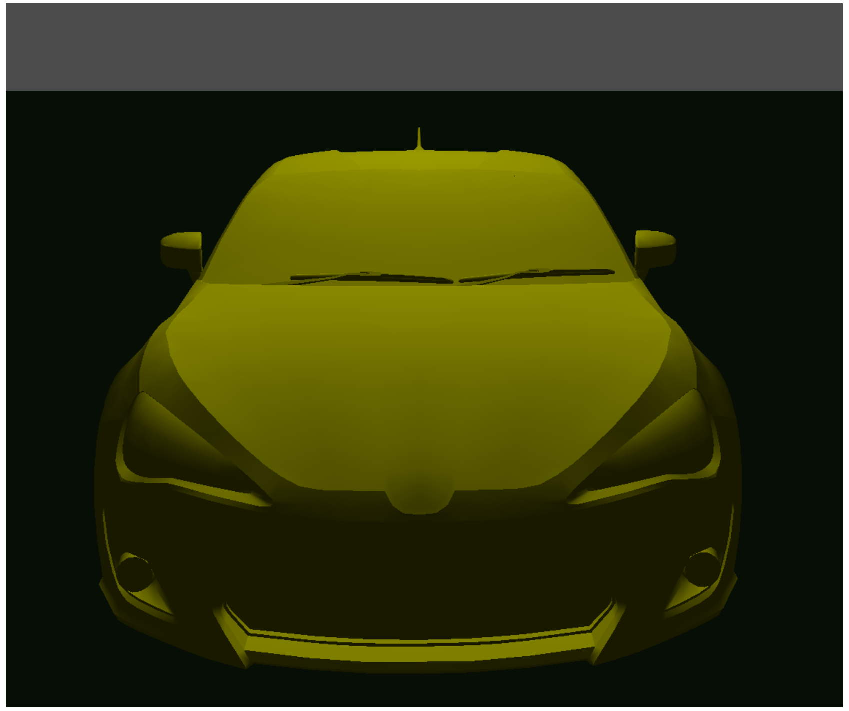
Ambient reflection

- Send the following uniforms to the shaders
  - (ambientLightColor * ambientLightIntensity)
  - reflectionColor
  - ambientReflectance (as uniforms) to the shaders.
- Following the equations seen in class, compute the ambient reflection in the vertex shader and send the color to the fragment shader.
- Can you see the car?

8.2 Gouraud shading 2 (per vertex light)

Diffuse reflection

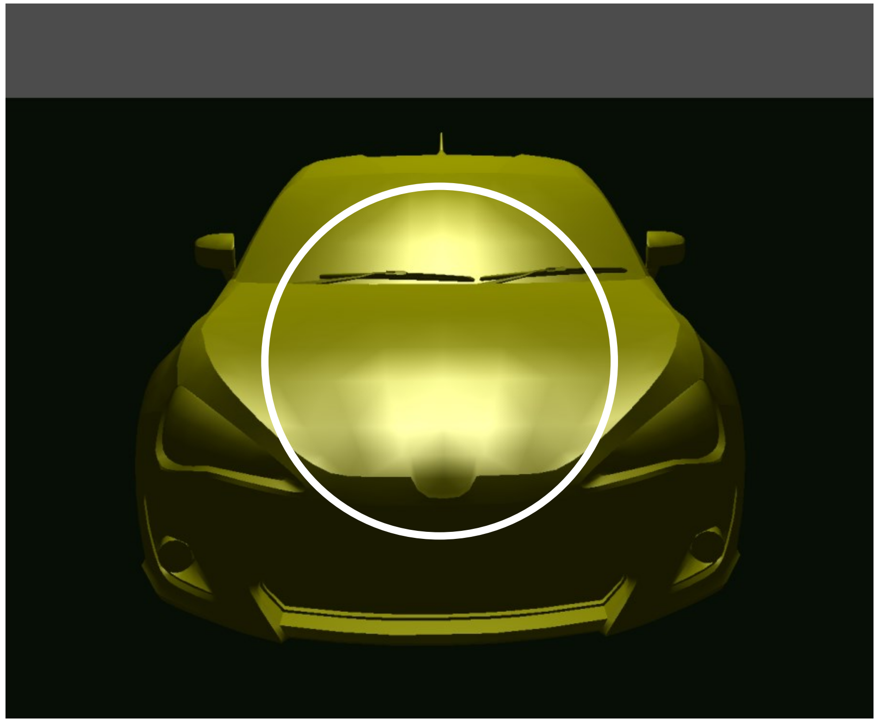
- Send the uniforms to the shaders
    - light1Position
    - light1Color * light1Intensity
    - diffuseReflectance
- Following the equations seen in class, compute the diffuse reflection in the vertex shader and add it to the final color.
    - Pay attention to the space of the point and vectors!
    - They should all be in the same coordinate space, in the shader I suggest using the eye space, aka camera space, but learnopengl do it in the world space.



8.3 Gouraud shading 3 (per vertex light)

Specular reflection

- Send the uniforms to the shader
  - light1Position
  - light1Color * light1Intensity
  - specularReflectance
  - specularExponent
- Following the equations seen in class, compute the diffuse reflection in the vertex shader and add it to the final color.
  - Pay attention to the space of the point and vectors!
  - They should all be in the same coordinate space, in the shader I suggest using the eye space, aka camera space, but learnopengl do it in the world space.

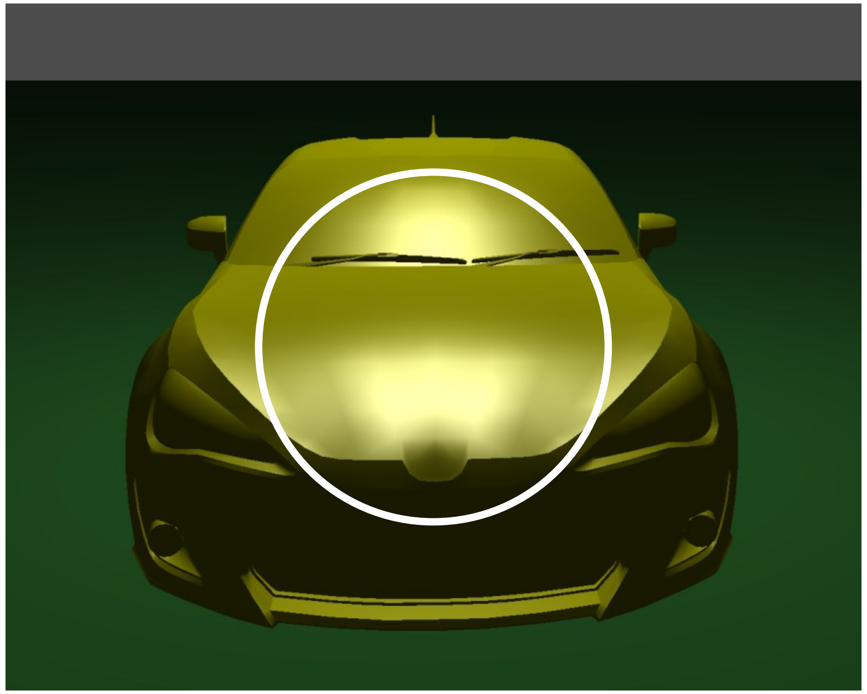


- Congratulations! You have completed a Gouraud shading implementation of the Phong reflection model!

8.4 Phong shading (per pixel light)

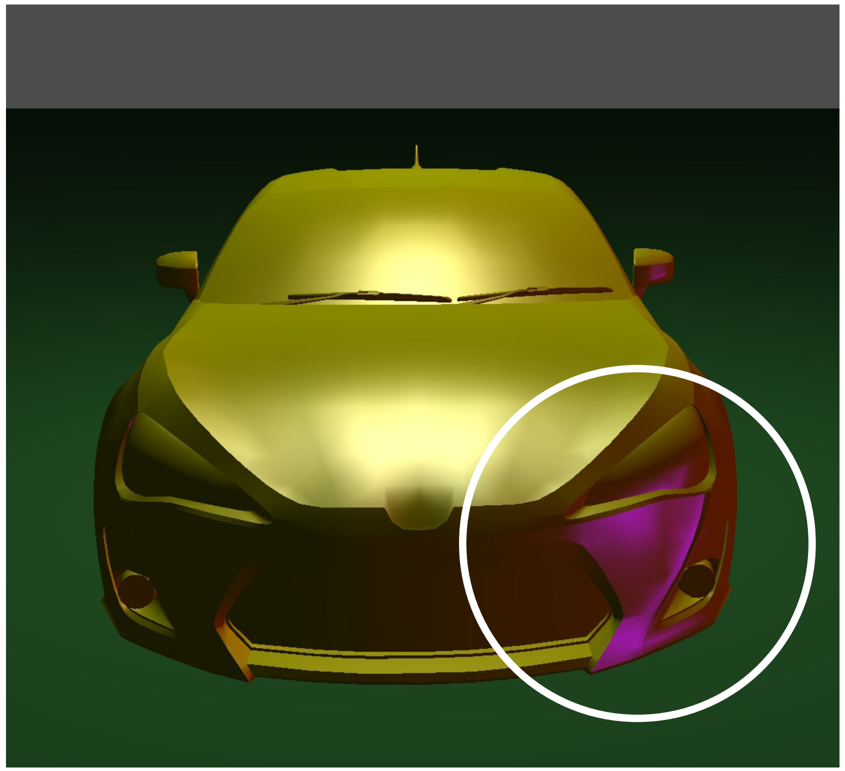
- Move light shading to fragment shader.
  - you will also need all the uniform variables from before in the fragment shader
  - You need to send the information about the normal and position of the vertex from the vertex shader to the fragment shader.
  - Try to apply all the space transformations in the vertex shader (and not in the fragment shader) since this will save in computations

- Instead of interpolated light values, now you should get a smooth light computation.
- Congratulations! This is the Phong shading implementation of the Phong reflection model!



8.5 Two light sources

- 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

- Send the uniforms to the fragment shader
  - attenuationC1
  - attenuationC2
  - attenuationC3
- Following the equations seen in class, compute attenuation using the distance of the vertex to the light source and the 3 attenuation parameters.