Computer Graphics

Aristotle University of Thessaloniki,
Faculty of Engineering

Assignment 3: Virutal Scene Photo

Konstantinos Chatziantoniou
8941
konstantic@ece.auth.gr

June 19, 2020

1 Code Structure

Demos:

- demo.m Script that demonstrates the shading of projected 3d object. It creates 8 images for each type of ligthing and their combination for the phong and gouraud shading algorithms.

Functions:

- ambientLight.m Calculates the colors of a point caused by ambient light.
- diffuseLight.m Calculates the colors of a point caused by diffusal light.
- specularLight.m Calculates the colors of a point caused by specular light.

  Implementation Note

  The above functions are "vectorized" specifically for a scanline of a triangle. They use the same point (triangle's center of mass) but different coefficients for each point of the scanline.

- findVertNormals.m Calculates the normlized vector, perpendicular to the surface of each points of the object. Since each point belongs to more than one triangles, the vectors perpendicular to the surface of each triangle are summed and then normalized.
- projectCameraKu.m Given the target's position and up vector for the camera calculates the x-y-z axis for the camera. Then calls projectCamera for the actual projection.

- shadeGouraud.m Applies gouraud shading to a single triangle. The color of the internal points of the triangle are calculated by interpolating the colors of the edges.
- shadePhong.m Applies phong shading to a single triangle. The light coefficients and normal vectors for each point are calculated by interpolating the coefficients and normal vectors of the edges. Then the color is calculated for each point using the interpolated values.
- photographObject.m Combines the above functions for coloring and the functions of the previous assignment for projection to implement the whole process of painting a projected 3d objects.

Implementation Note

Most of the files contain internal functions, like colorInterp, implemented in the previous assignments and modified for this one.

Information about the input and ouput of the functions can be found as comments in each file.

2 Implementation Explanation

Gouraud shading

First we calculate the colors of the 3 edges of the triangle using the scalar versions of the ambientLight.m, diffuseLight.m,specularLight.m functions. Then we calculate the x coordinate of active edges using the slopes and the colors using a gradient. Finally the colors of the scanline are calculated by interpolating the colors of the active edges.

Phong shading

Instead of calculating a gradient for the color, we calculate the gradients of the ligthing coefficients and the normal vectors. By modifying the colorInterp function, we apply the interpolation to the coefficients instead of the colors. Finally, the colors of the scanline are calculated using the vectorized version of the ambientLight.m, diffuseLight.m,specularLight.m functions.

Lighting

The implementation of the ambientLight.m, diffuseLight.m,specularLight.m functions is a straigtforward application of the mathematical formulas, taken from the notes.

For diffuse light, we can check the final light intensity, and change all the negative values to zero. But for specular light, doing so is wrong. If the *ncoeff* is even, the negative cosines will give positive final light intensity. So we have to perform the check on the cosine before the application of the *ncoeff*.

$cosab = cosab. * (cosab > 0)$;

Rest of the code

Every internal function has already been explained in the previous assignments

- `https://github.com/KonstantinosChatziantoniou/ComputerGraphics-TriangleFilling`
- `https://github.com/KonstantinosChatziantoniou/ComputerGraphics-TransformationProjectior`

3 Assumptions

We assume that the 3 points-edges of the triangle in the 'F' array, are correctly ordered to infer the orientation of the triangle.

Also, when calculating the diffusal and specular light effect, we don't take into account the shadow cast by the object, that could potentially be cast on itself.

4 Results

The demo was executed on an i3 4170 @ 3.7Ghz. The whole process with gouraud shading took **4 sec** and with phong shading took **30 sec**.

Gouraud Shading

Figure 1: Gouraud Shading. ONLY ambient light

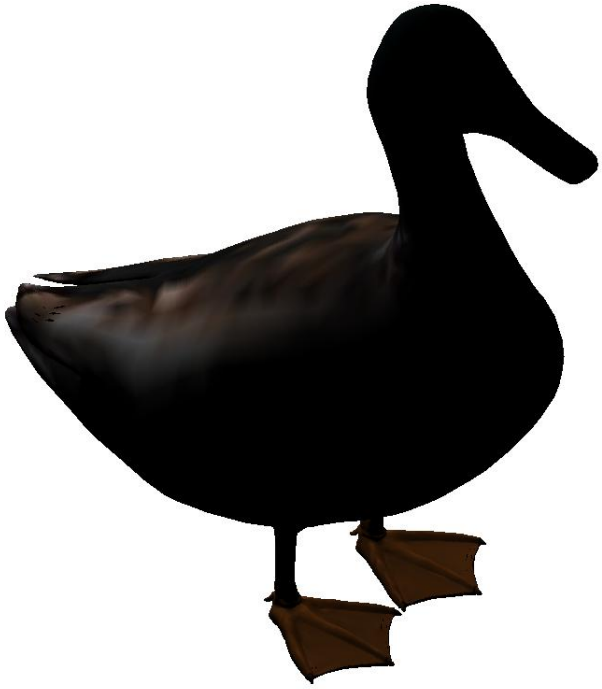

Figure 2: Gouraud Shading. ONLY diffuse light

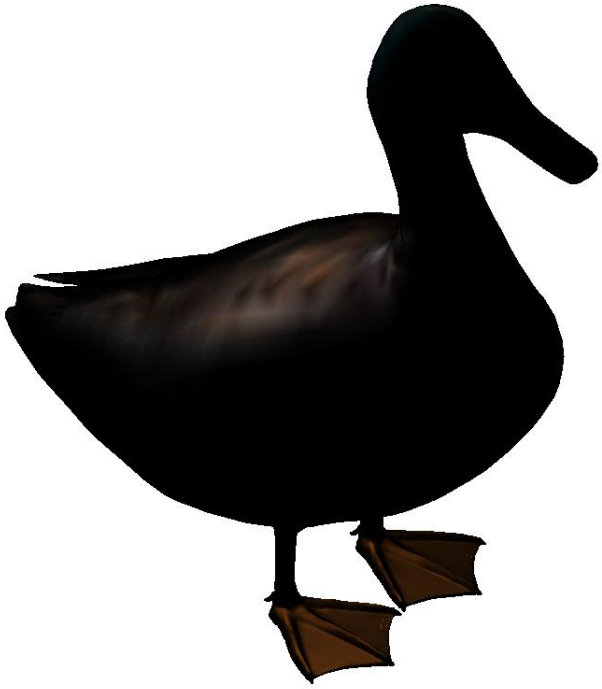

Figure 3: Gouraud Shading. ONLY specular light

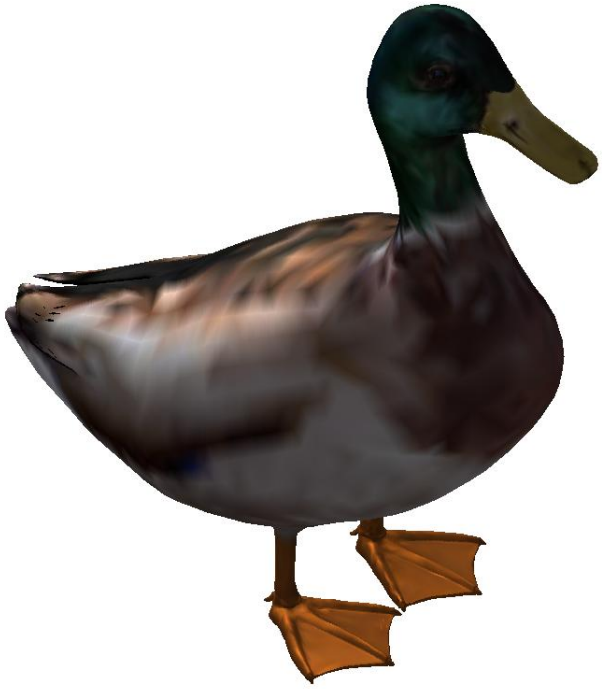

Figure 4: Gouraud Shading. Complete lighting.

Phong Shading

Figure 5: Phong Shading. ONLY ambient light

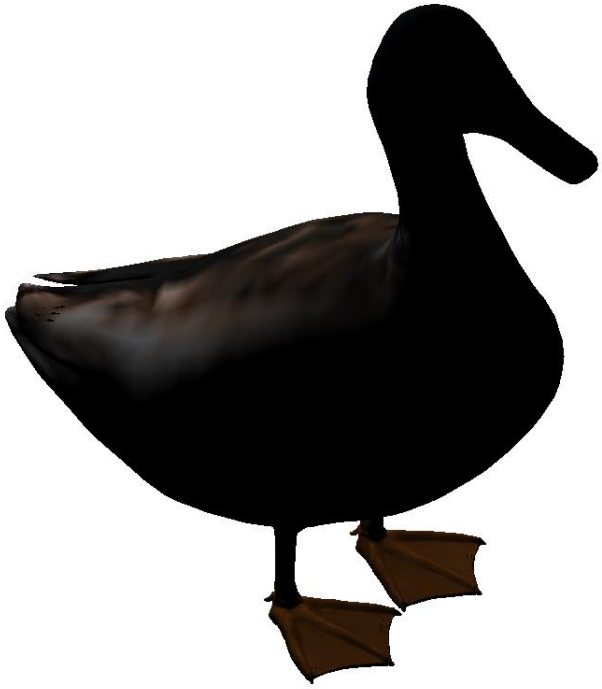

Figure 6: Phong Shading. ONLY diffuse light

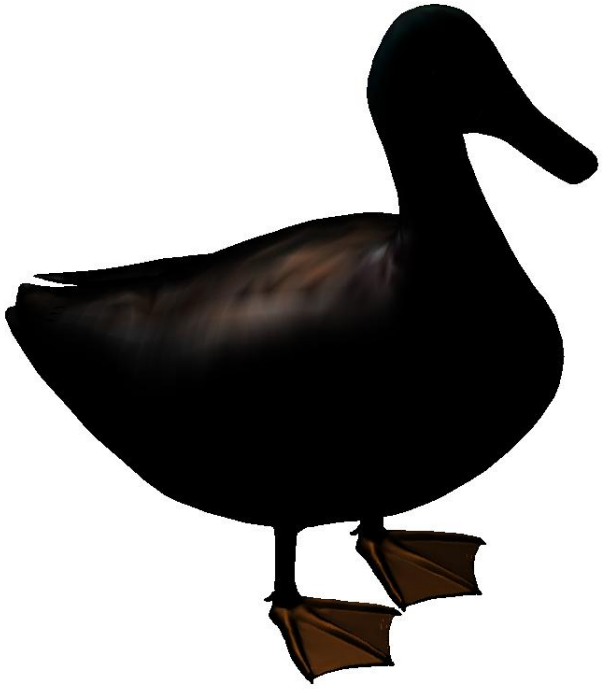

Figure 7: Phong Shading. ONLY specular light

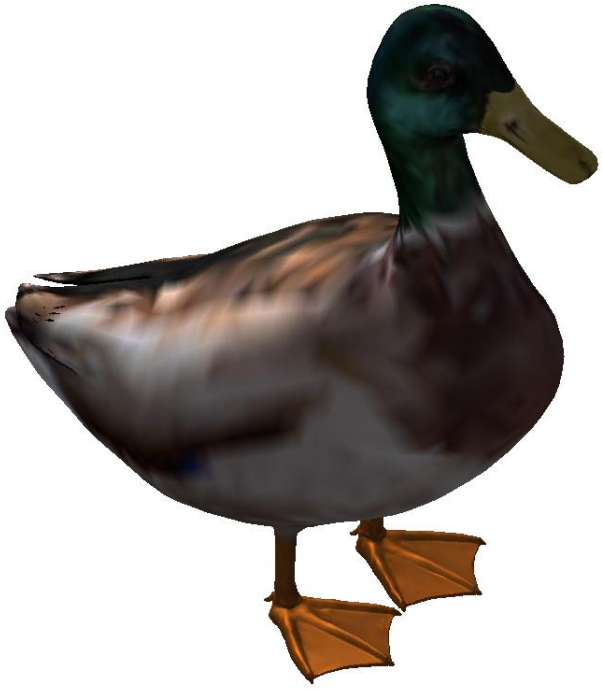

Figure 8: Phong Shading. Complete lighting.

Comments

The diffusal and specular light intensity is high at the same parts of the duck, with the specular light being brighter at the center of those parts and getting dimmer the further away.

The Phong and Gouraud algorithms, seem to produce the same results.

Figure 9: Absolute difference between Phong and Gouraud.

The difference is barely visible and doesn't explain the x8 execution time. In figure 9 we can distinguish the outline of the duck's feet and some blur lines on the feathers.

The advantage of Phong is apparent when we have a moving/rotating object and the transition of specular light is smooth.