Programming Assignment 2

COMP 575/770 Spring 2016

Due: Wednesday, March 23, 2016

Instructions

- Please work on the assignment on your own. It is okay to discuss the assignment with other students, but please write your own code independently. If you use code from the Internet or any other source, please acknowledge the source.
- You are free to use any programming language you are comfortable with, but try not to use anything too obscure. C/C++, C#, Objective-C, Java, and Python are common choices.
- You will need to use a fairly small amount of OpenGL and GLUT to get your image on-screen.
- Submit your source code via email along with a README file containing compilation instructions, additional required libraries (if any), and a short description of what the different parts of your program do.
- You may use any third-party library for basic vector algebra and matrix algebra (in particular, for matrix addition, subtraction, multiplication, transpose, determinant, or inverse). You must, however, construct all the matrices on your own.
- You are not allowed to use OpenGL or Direct3D to perform hardware rasterization. You must implement your own transformation pipeline, rasterizer, and shading algorithms. Only use OpenGL/GLUT or any other library to display the final image.
- Submit your source code via email along with a README file containing compilation instructions, additional required libraries (if any), and a short description of what the different parts of your program do.
  Email Ids: Tanmay (tanmay@cs.unc.edu), Michael (mkcolema@cs.unc.edu)

Objective The goal of this assignment is to write a simple rasterization pipeline and use it to draw a single shaded sphere, which is in turn represented using multiple triangles. These triangles (and their vertices) are generated using code that is provided on the course webpage. For reference, some views of the scene are shown below.

1. **Transformations and Rasterization**
   Use the following sequence of transformations to display the sphere:

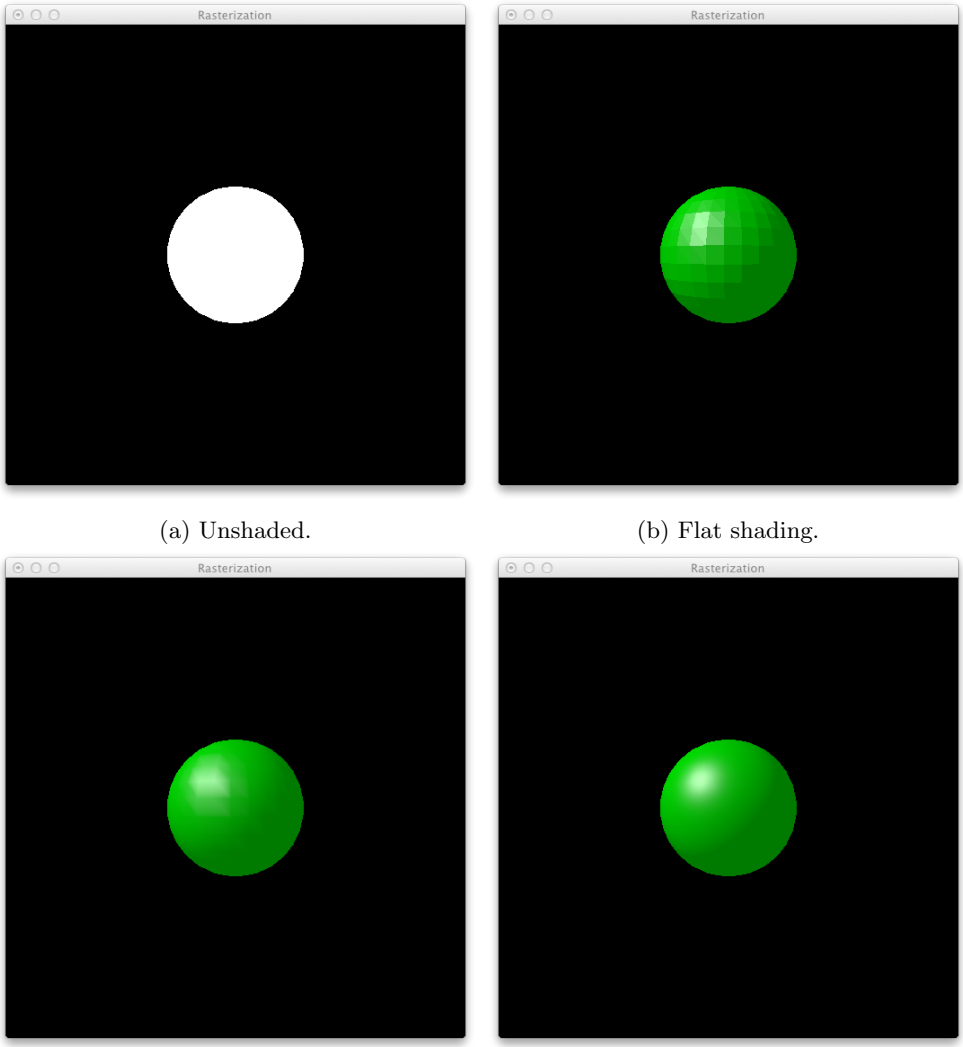
   (a) A function in C++ to generate triangles for a unit sphere has been provided. Use a modeling transform to transform the unit sphere to a sphere centered at $(0, 0, -7)$ with a radius of 2.

   (b) Use a camera transform with eye point at $\mathbf{e} = (0, 0, 0)$, and orientation given by $\mathbf{u} = (1, 0, 0)$, $\mathbf{v} = (0, 1, 0)$ and $\mathbf{w} = (0, 0, 1)$. (Note that the camera is looking along the direction $-\mathbf{w}$.)

   (c) Use a perspective projection transform with $l = -0.1$, $r = 0.1$, $b = -0.1$, $t = 0.1$, $n = -0.1$, $f = -1000$.

   (d) Use a viewport transform with $n_x = 512$, and $n_y = 512$.

   After transformations, you may use any algorithm to rasterize the triangles. Back-face culling is optional, but you must use a depth buffer to handle occlusion.



(a) Unshaded. (b) Flat shading.

(c) Gouraud (per-vertex) shading. (d) Phong (per-pixel) shading.

Figure 1: Four views of the scene used for this assignment.

2. **Shading**
   Now use the following material properties to shade the sphere: $k_a = (0, 1, 0)$, $k_d = (0, 0.5, 0)$, $k_s = (0.5, 0.5, 0.5)$, $p = 32$. Use an ambient light intensity of $I_a = 0.2$. Assume a single point light source at $(-4, 4, -3)$, emitting white light with unit intensity and no falloff. Implement (a) flat shading (evaluating the shading at the centroid of each triangle), (b) Gouraud shading, and (c) Phong (per-pixel) shading. Apply gamma correction with $\gamma = 2.2$ in each case.