Programming Assignment 1

COMP 575/770 Spring 2016

Due: February 15, 2016

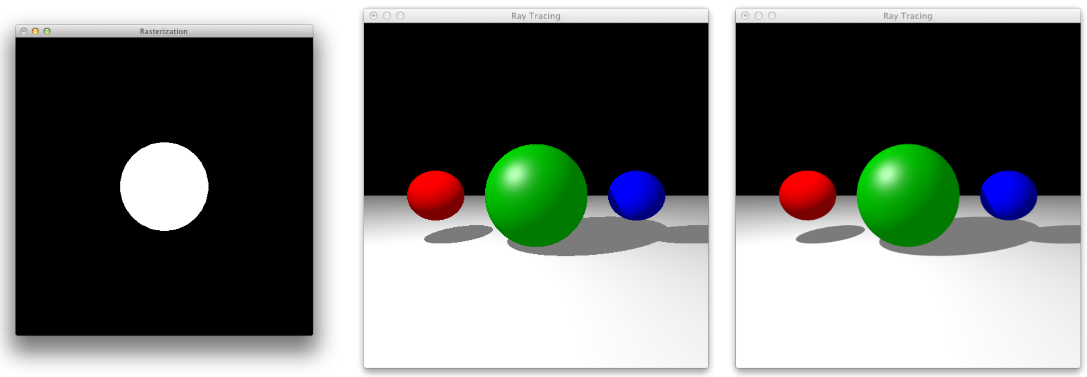
Instructions

- 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. A comprehensive tutorial for GLUT is available at:
  http://www.lighthouse3d.com/tutorials/glut-tutorial/
  This tutorial is Windows-oriented.
  A tutorial on starting an OpenGL project in Visual Studio is here:
  http://mycodelog.com/2015/10/08/opengl-freeglut-in-visual-studio-2015/
  You can find out about compiling GLUT code under OSX at:
  http://blog.onesadcookie.com/2007/12/xcodeglut-tutorial.html
  and under Linux at:
  http://www.its.monash.edu.au/staff/systems/linux/technical/gluthome.html
  A brief tutorial for Python is also available at:
  http://code.activestate.com/recipes/325391-open-a-glut-window-and-draw-a-sphere-using-pythono/
  Tutorial for OpenGL
  http://www.opengl-tutorial.org/beginners-tutorials/tutorial-1-opening-a-window/
  A function that would be useful is :
  https://www.opengl.org/sdk/docs/man2/xhtml/glDrawPixels.xml
- 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 ray tracer which can render a simple scene consisting of a plane and three spheres, and shade them using the Phong model. For reference, some views of the scene are shown below.

1. **Ray Intersection**
   The scene consists of the following four objects:
   (a) Plane P, with equation $y = -2$.
   (b) Sphere S_1, with center at $(-4, 0, -7)$ and radius 1.
   (c) Sphere S_2, with center at $(0, 0, -7)$ and radius 2.
   (d) Sphere S_3, with center at $(4, 0, -7)$ and radius 1.

   Assume a perspective camera, 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 $-\mathbf{w}$.) Assume



(a) Intersections only. (b) With shading and shadows. (c) With antialiasing.

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

that the viewing region on the image plane is defined by $l = -0.1$, $r = 0.1$, $b = -0.1$, $t = 0.1$, and $d = 0.1$. Also assume that the image resolution is 512×512 (i.e., $n_x = n_y = 512$).

Write a ray tracer which generates eye rays through the center of each pixel, and calculates the intersection between each ray and each object in the scene, also calculating the closest intersection (if any) for each ray. For each pixel, if the corresponding ray intersects an object, set the pixel's color to white; otherwise, set the pixel's color to black. See Figure 1a for a reference image.

2. **Shading**
   Use the Phong model to shade the scene, with the following material parameters for each object:

   (a) P: $k_a = (0.2, 0.2, 0.2)$, $k_d = (1, 1, 1)$, $k_s = (0, 0, 0)$, with specular power 0.
   (b) S_1: $k_a = (0.2, 0, 0)$, $k_d = (1, 0, 0)$, $k_s = (0, 0, 0)$, with specular power 0.
   (c) S_2: $k_a = (0, 0.2, 0)$, $k_d = (0, 0.5, 0)$, $k_s = (0.5, 0.5, 0.5)$, with specular power 32.
   (d) S_3: $k_a = (0, 0, 0.2)$, $k_d = (0, 0, 1)$, $k_s = (0, 0, 0)$, with specular power 0.

   Note that k_a, k_d, and k_s values are specified as (red, green, blue) values. Assume a single point light source at $(-4, 4, -3)$, emitting white light with unit intensity and no falloff.

   Modify your ray tracer to store at each pixel the appropriate illumination values computed using the Phong shading model. Also model shadows from the point light using shadow rays. Perform gamma correction with $\gamma = 2.2$. See Figure 1b for a reference image.

3. **Antialiasing**
   The image produced in the previous part of the assignment contains unsightly "jaggies" at the boundaries of the spheres. In this part, we will remove the jaggies using antialiasing. Generate $N = 64$ samples of the image within each pixel, by generating N random eye rays within each pixel. For each pixel, apply a box filter with amplitude $1/N$ and support equal to the extents of the "pixel rectangle" to estimate the pixel value. The resulting image should have significantly reduced aliasing artifacts. See Figure 1c for a reference image.