CSC 433 / 533: Spring 2015
Assignment 1

Assigned: Monday, Feb 2nd
Due: 11:59pm Friday, Feb 13th

For this assignment you will write a single program that uses OpenGL and renders objects in normalized device coordinates. The programs will be graded on the linux workstations on the 9th floor of Gould-Simpson building.

In this program you will work with two shader programs and three geometric objects requiring multiple vertex array objects and vertex buffer objects. And, you will use GLUT to respond to keyboard presses.

For all programs this semester:

1. Provide a Makefile that builds the program(s). We will not grade programs that do not compile.
2. Your directory should be self-contained and not need any files from other directories. So, it will contain all source code for both the CPU and GPU, and it will include any required data files.
3. Include a readme.txt file and include any special instructions or assumptions.

Task:

1. Extend the code for example 1 from OpenGL Programming Guide (red book) to add the following capabilities. Control of the program is through keyboard input, and below is a description of which keys direct actions.
    a. Create the geometric description of two additional objects: a single triangle with different colors at each of the three vertices and a set of triangles approximating a circle.
        i. For the single triangle, you can either 1) create a buffer of vertex locations and a separate buffer for vertex colors or 2) use a single buffer for both vertex locations and colors. Center this triangle in the screen area. Put a single vertex at the top of the triangle and color it red. Set up the base of the triangle with the left vertex colored blue and the right vertex colored green. The book *OpenGL Shading Language Cookbook Edition 2* in Chapter 1 in the section "Sending data to a shader using vertex attributes and vertex buffer objects" has an example of using two buffers. The OpenGL Programming Guide has an example of using a single buffer in chapter 4 on pages 150 and 152. Become familiar with both approaches.
        ii. For the circle, create a buffer of vertex locations. A good way to generate vertices along the perimeter of a circle is with a parametric equation, and in this case the parameter is an angle. For this parametric approach you can use the equations:

x = radius * sin (angle) and y = radius * sin (angle)

and as the angle ranges from 0 to 2π you can produce the x,y values around the perimeter of the circle.

iii. For the circle, this is a time that you could use a triangle fan. While this isn't required, you might find it interesting and easier.
iv. Put the description of these objects into vertex array objects and vertex buffer objects for drawing.
v. All x and y coordinates are in normalized device coordinates which range between -1 and 1.

b. Modify the existing vertex shader from Example 1 to use a uniform variable for the color of the two triangles and the circle. This uniform variable can have different colors assigned through keyboard input.
    i. Create a second shader program for the single triangle which will use both vertex locations and a different color at each vertex.
c. Rendering order: always draw the two triangles first, then the single triangle and finally the circle.
d. Create a square window for OpenGL rendering.
e. Keyboard commands processed with GLUT. For these keyboard commands, the windows focus needs to be in the OpenGL display window, but then any associated values, read from stdin, will need focus in the command line window associated with the OpenGL application.
    i. c – specify the color for both the two triangles and the circle. When processing this command, the user needs to provide three floating point numbers through stdin for the color, in the range of 0. to 1., in the order of red, green, and blue components. The default initial color is blue (0., 0., 1.)
    ii. s – shaded surface display : render the objects as solid surfaces. Control of the drawing or rendering mode in OpenGL mode is controlled through a call to glPolygonMode. After changing the OpenGL rasterization (drawing) mode, you can force a redraw through GLUT using glutPostRedisplay function call.
    iii. w – wireframe display : render the objects in wireframe. You can use the glLineWidth function to specify the thickness of lines.
    iv. g – generate the geometry for the circle. When processing this command, the user needs to provide a floating point number for radius value in the range of 0. to 1., and a integer number of steps to use when generating triangles to approximate the circle.
    v. x – toggle display of the two triangles, default value is on.
    vi. y – toggle display of the single triangle, default value is on.
    vii. z – toggle display of the circle, default value is off. If a circle has not been generated through the 'g' key, then there will be no circle to display.
    viii. q or ESC key – quit application
f. Include error checking for input values and write information about errors to stdout.
g. In your makefile create an executable named program1.

2. Grading rubric – 100 points total

a. [20 points] rendering of the three geometric objects: 1) two triangles, 2) single triangle and 3) circle in the defined order and with the correct appearance.

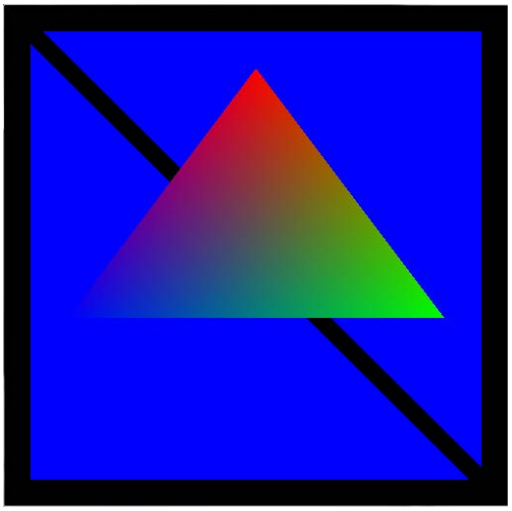
b. [20 points] change the color of the two triangles and circle with the ‘c’ key to specify the new color specified with a R,G,B triple of floating point numbers and a uniform variable in the shader program.

c. [15 points] switch between wireframe and surface display for all three objects, using the ‘s’ key to set the OpenGL state for solid rendering and the ‘w’ for wireframe rendering.

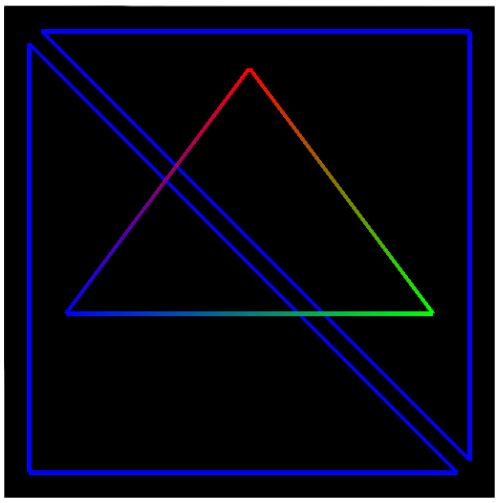
d. [25 points] modify the approximation of the circle geometry in response to keyboard command by using the ‘g’ key and then providing a radius and number of steps around the circle.

e. [20 points] successful toggle on and off the display of the three geometric objects using the ‘x’, ‘y’, and ‘z’ commands.

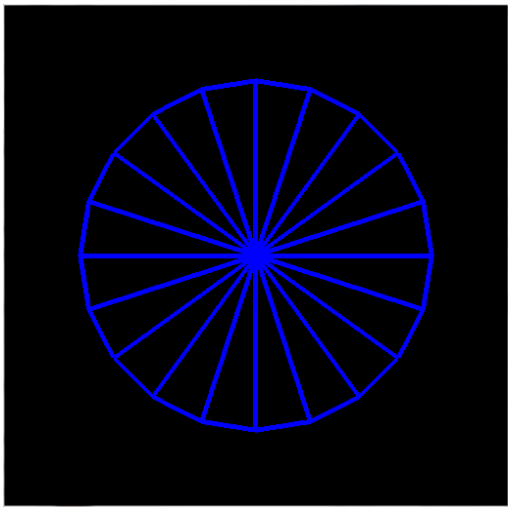
Sample output



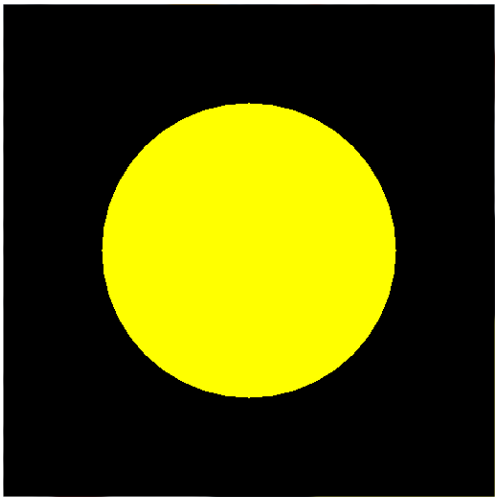
initial screen



wireframe display
(linewidth = 5)



Circle radius 0.7 steps 21



Circle radius 0.6 steps 41

c 1 1 0

Submission Instructions

Submit your files on the host **lectura.cs.arizona.edu** using the command
turnin cs433s15-assg1 [files] We will use your makefile to create the executable, and will test the resulting executable.

Assignment Advice

1. Start early.
2. Do your own work.
3. Check piazza regularly.