CSC 433 / 533: Spring 2017
Assignment 1

Assigned: Tuesday, January 24th
Due: 11:59pm Tuesday, Feb 7th

For this assignment you will work with a single program that uses OpenGL and renders objects in normalized device coordinates, $-1 \leq x \leq 1$ and $-1 \leq y \leq 1$. The programs will be graded on the Mac workstations on the 9th floor of Gould-Simpson building.

In this program you will work with two shader programs and multiple geometric objects requiring multiple vertex array objects and vertex buffer objects. And, you will use GLFW for the window support and user interface (UI), including responding to keyboard presses for this assignment.

For all programs this semester:

1. Provide a Makefile that builds the program. We will not grade programs that do not compile.
2. Your directory must contain all the files needed to build your executable. Your Makefile will include the directives to use the GLFW headers and libraries and and gl3w.c for the latest OpenGL routines available in OpenGL 4.1. On the Mac OS X workstations the GLFW files are in standard system location. Your submission will contain all source code for both the CPU and GPU, and it will contain 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 three additional objects: 1) a single triangle with different colors at each of the three vertices, 2) a circle and 3) a heart curve. For the circle and the heart curve, use lines to approximate the parametric shape. You will end up with two objects described with triangle(s) and two objects described with lines.
        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 4 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, one for vertex position and the other for vertex color. The OpenGL Programming

Guide has an example of using a single buffer in chapter 4 in the section *Color and OpenGL, subsection Vertex Colors*. 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 the circle is with parametric equations, and in this case the parameter is an angle. For this parametric approach you can use the equations:

$$x = r * \cos(\text{angle}) \text{ and } y = r * \sin(\text{angle})$$

and as angle ranges from 0 to 2π you can produce the (x,y) values around the perimeter of the circle. To create this curve, the user will provide two parameters, the radius,r and the number of steps. Create vertices and draw them as a triangle fan (GL_TRIANGLE_FAN).

iii. For the heart curve (http://mathworld.wolfram.com/HeartCurve.html) we're using the curve described with parametric equations,

$$x = 16/17*\sin(\text{angle})*\sin(\text{angle})*\sin(\text{angle})$$
$$y = 1/17*(13*\cos(\text{angle}) - 5*\cos(2*\text{angle}) - 2*\cos(3*\text{angle}) - \cos(4*\text{angle}))$$

where I've added a scale factor to keep the values in NDC. As the parameter, angle, ranges from 0 to 2π the values of a curve will be computed. To create this curve, the user only needs to provide the number of steps. Draw these vertices with line strips (GL_LINE_STRIP).

iv. Put the description of these objects into buffer objects for drawing.

v. All x and y coordinates are in normalized device coordinates which range between -1 and 1. Choose parameters that comply with this restriction. Otherwise the values will be clipped away by the GPU pipeline.

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, then the circle, and finally the heart curve.

d. Create a square window for OpenGL rendering of a size 800,800.

e. Keyboard commands processed with GLFW. 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 the two triangles and the parametric curves. 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 polygonal objects as solid surfaces. Control of the drawing or rendering mode in OpenGL is controlled through a call to glPolygonMode. After changing the OpenGL

rasterization (drawing) mode, you can force a redraw through GLFW using glfwSwapBuffers function call.

iii. w – wireframe display : render the polygonal 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 one floating point number for radius in the range of 0. to 1., and an integer number of steps to use when generating triangles to approximate the circle.

v. h – generate the geometry for the heart curve and one additional parameter which is the integer number of steps to use when generating the heart curve.

vi. x – toggle display of the two triangles, default value is on.

vii. y – toggle display of the single triangle, default value is on.

viii. z – toggle display of the circle, default value is off.

ix. a – toggle display of the heart curve, default value is off. If this curve has not been generated through the 'h' key, then there will be no heart curve to display.

x. 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 four geometric objects: 1) two triangles, 2) single triangle, 3) heart curve and 4) circle in the defined order and with the correct appearance.

b. [15 points] change the color 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.

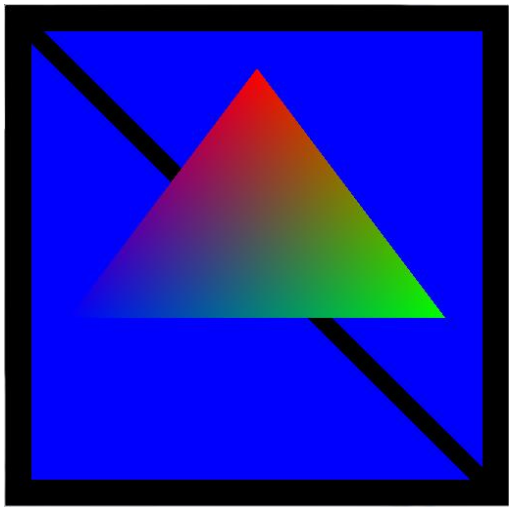
d. [15 points] modify the approximation of the circle geometry in response to keyboard command by using the 'g' key.

e. [15 points] modify the approximation of the heart curve geometry in response to the keyboard command by using the 'h' key.

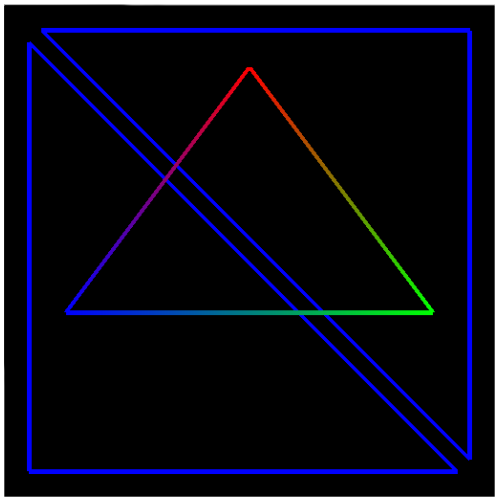
f. [10 points] successful toggle on and off the display of the geometric objects using the 'a, 'x', 'y', and 'z' commands.

g. [10 points] choose a desktop / laptop first-person shooter game and describe the keyboard and mouse actions used to move through environment. It does not have to be a modern game, but don't select a game where a game controller is used. For example, how do you *move forward*, how do you *turn your gaze to the right or left*, how do you *move up and down, etc.* Describe as many of the actions as you can about how to modify your viewing of the environment. Put your answer to this question in the readme.txt file included in your submission.

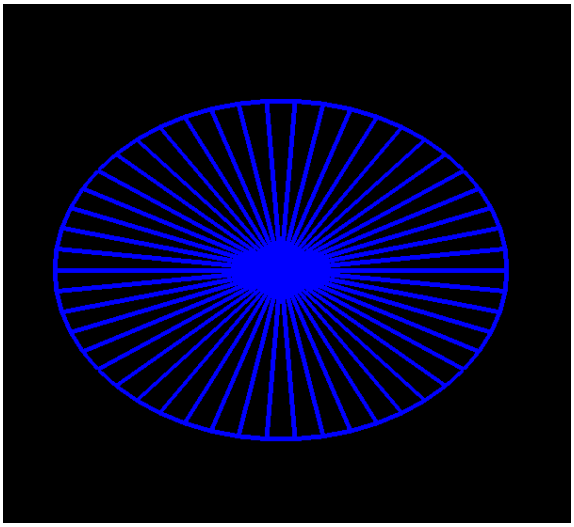
Sample output



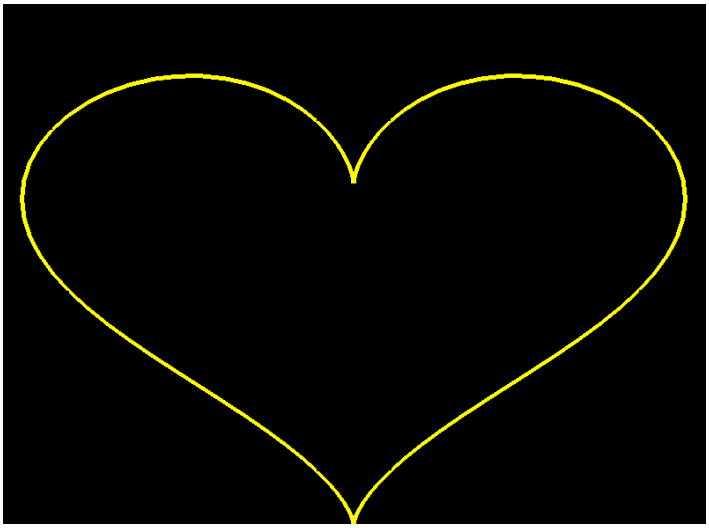
initial screen
shaded display



initial screen
wireframe display
(linewidth = 5)



Circle radius .6
51 steps
C 0 0 1



Heart curve
101 steps
c 1 1 0

Submission Instructions

Submit your files while on the host **lectura.cs.arizona.edu** using the command **turnin cs433s17-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.