Math 155A - Introduction to Computer Graphics – Winter 2017

Instructor: Sam Buss, Univ. of California, San Diego

Project #1 – Create two tetrahedra with smooth and flat colors.

Due date: Friday, January 20 9:00pm.

Goals: Gain some basic familiarity with triangle fans and triangle strips. Learn how to make triangles of solid color as well as how to shade colors smoothly. Learn how to use key controls to control viewpoint, and toggle wireframe.

What to hand in: When you are done, place your C++ files, executable, and Visual Studio solution together in a separate folder in your PC computer account in the APM basement labs. If you were able to create a suitable image file showing z-fighting (PNG or JPEG or BMP files are all OK), please also place this in the same folder. There should be nothing in this folder except your files for this homework assignment, and the creation/modification dates should match the turn in date. The program must compile and run on these computers.

Grading will be personalized and one-on-one with one of the TAs Srivastava or Wang or with Sam Buss. Your program must run on the PC lab, you must come into the PC lab and meet one of us. You will have to show your source code, run the program, make changes on the spot to your program and recompile as requested by the grader, and be able to explain how your program works and why it renders what it does. The grading should be completed promptly, but no later than the due date for the next programming assignment.

FOR PROJECT #1, PLEASE DO THE FOLLOWING STEPS #1 - #11.

1. **Download** the Tetrhedra program from the zip file TetrahedraProject.zip. Extract these into a directory named Tetrahedra. Be sure to share this in your networked, permanent folder. Files left elsewhere on the computer lab computers are likely to be to erased automatically! (The full URL for the zip file is: http://www.math.ucsd.edu/~sbuss/CourseWeb/Math155A_2017Winter/Project1/TetrahedraProject.zip.)
2. There is an executable "**TetrahedraDemo.exe**" that shows (more-or-less) **how your program should end up**. This is a 64-bit version and should run on the computer lab. If you are using another system, you may also try the 32-bit version, TetrahedraDemo_32bit.exe.
   Experiment with this program. Notice the following items and keyboard commands.
   a. There are two tetrahedra shown.
   b. The arrow keys (left, right, up down) control the view position. The scene moves only when you press an arrow key. (No animation yet.)
   c. If the rate of movement is too slow, press "R" to make it faster. If it is too fast, press "r" to make the rate slower.
   d. The "w" key toggles wire-frame mode. Back faces are culled.
   e. The shape on the left is smooth shaded, (GL_SMOOTH option). The shape on the right is flat-shaded (GL_FLAT option).
   f. The colors on the top three sides of the flat shaded tetrahedron are fully saturated, for example, the red is ⟨1,0,0⟩, the green is ⟨0,1,0⟩, etc. The color on the bottom face is grey.

g. Similar numeric values are used for colors on the smooth shaded shape. However, the top vertex is colored white, and grey is not used.

3. **Find the source files "Tetrahedra.cpp" and "Tetrahedra.h" in TetrahedraBase and use these to create a Visual C++ project. Examine the source code and run this program.** This program looks somewhat like the TetrahedraDemo.exe. However, it draws cubes instead of the tetrahedra. When you examine the source, do the following:
    a. Examine the code and understand how the cubes are specified (using GL_QUADS and as GL_QUAD_STRIP).
    b. Examine the code to understand the OpenGL commands that control the shading (flat or smooth).
    c. Examine the code to understand the OpenGL commands that turn wire frame mode off and on.
    d. Understand how OpenGL enables back face culling.
    e. Find every place in the code where glutPostRedisplay is called. Understand why it is called in these places.
4. **Re-write the code for the shape on the right.** In place of the code given for the cube on the right, write code that draws the shape shown in TetrahedraDemo. Use one triangle fan plus one additional triangle to build the shape.
   For this Tetrahedron, rewrite the routine MyDrawColorShape1(). The tetrahedron should
    a. **Have each side length equal to sqrt(3).**
    b. Be centered at the origin. (The routine drawScene takes care of moving to an appropriate place on the screen.)
    c. Match the colors in the Demo program.
5. **Re-write the code for the shape on the left.** In place of the code given for the cube on the left, write code that draws the shape shown in TetrahedraDemo. This time, use one triangle strip. That is all four faces are given in a single triangle strip. Again, match the appearance in the demo program.
6. Be sure that **all faces of the shapes are facing in the correct outward direction**. Be careful about specifying vertices in the right order. There should not be any holes in your tetrahedra. It is OK to use **glFrontFace(...)** if you wish to.
7. **Implement a new keyboard control "c" to toggle back face culling off and on.** Can you see the difference in wireframe mode? Can you see it in solid mode?
8. **Understand the difference between flat and smooth shading.** Be able to discuss the differences with the TA grading your program, and to explain how the shading is caused by the source code.
9. **Examine carefully the way the program works in wire frame mode.** Do you notice anything unusual as the tetrahedra are rotated in wire frame mode? Are there any artifacts due to ***aliasing***? (E.g., jagged lines, or crawling visual artifacts.) Can you see any ***z-fighting***? (See item 10.) These effects may be very subtle. If you have trouble seeing aliasing problems, try slowing down the motion by pressing "r" a few times, then hold down arrow in key and watch the edges of the tetrahedra. Try this in both wire-frame mode and non-wire-frame mode. For z-fighting, use only wire-frame mode. Be ready to discuss what you see with your TA when you are being graded. (These phenomena may well be different on different machines! There is a tendency for OpenGL implementations of lines and of wire-frame mode to have small bugs and this may be part of what you see.) **But you may not be able to create z-fighting on a home computer!**
10. **If you are able to see any z-fighting:** Make a screenshot showing the z-fighting and save the image as PNG or BMP of JPEG file. (If using JPEG make sure it shows enough detail to show the z-fighting. z-fighting will be visible **only** in wire-frame mode, and **only** on the flat-shaded tetrahedron. It will show one of the edges having a few pixels replaced

with a different color (due the fact that the same edge is drawn twice, with different colors). The best way to find the z-fighting is to make the image large, slow down the rate ("r" command), use the arrow keys to rotate, and watch carefully for pixels flashing different colors. The lab computers are usually able to show z-fighting, but other computers (with different implementations of OpenGL) might not.
To form the screenshot on a PC, press "ALT" and the "RIGHT-SHIFT" and the "PRINT SCREEN" (PRT-SCN) buttons together. Then open an image program such as Paint, paste the screenshot image in, and save to disk.

11. **Turn in the project as described above.**

Program grading: Scale of 0 to 10. Personal grading session with a TA or the professor.