CSCE 4813 – Programming Project 4
Due Date – 03/29/2019 at 11:59pm

1. Problem Statement:

The purpose of this programming project is to implement a "toolbox" of graphics functions that might be of use in some future graphics program you create. These functions are grouped in three categories: geometric operations, clipping operations, and shading operations. The functions you must implement are:

Geometric operations:
- Translate point (x,y,z) by (Tx,Ty,Tx) and return (x',y',z')
- RotateX point (x,y,z) around X axis by theta and return (x',y',z')
- RotateY point (x,y,z) around Y axis by theta and return (x',y',z')
- RotateZ point (x,y,z) around Z axis by theta and return (x',y',z')
- Scale point (x,y,z) by (Sx,Sy,Sx) and return (x',y',z')
- Project point (x,y,z) onto Z=d plane and return (Xp,Yp)

Clipping operations:
- Calculate and return Cohen-Sutherland outcode for point (x,y,z)
- Accept line segment based on two outcodes (return T/F)
- Discard line segment based on two outcodes (return T/F)
- Clip line segment based on two outcodes (return T/F)

Shading operations:
- Normalize length of Ax,Ay,Az and return Bx,By,Bz
- Calculate DotProduct of Ax,Ay,Az and Bx,By,Bz and return scalar
- Calculate CrossProduct of Ax,Ay,Az and Bx,By,Bz and return Cx,Cy,Cz
- Calculate DiffuseTerm using Lx,Ly,Lz and Nx,Ny,Nz and return scalar
- Calculate IdealReflection using Lx,Ly,Lz and Nx,Ny,Nz and return Rx,Ry,Rz
- Calculate SpecularTerm using Vx,Vy,Vz and Rx,Ry,Rz and P and return scalar

2. Design:

Your first design task is to select appropriate names for all of the functions above and decide what input/output parameters are needed. You may want to create a very simple data structure to store the (x,y,z) coordinates of points or vectors to simply parameter passing. Remember the focus is on the functions above, so don't get carried away with fancy operator overloaded point or vector classes.

Your second design task is to work out the exact formulas needed to perform all of the operations above. You are welcome to look online or in the class notes to find what you need. As you know, there are fancy ways to do geometric operations using homogeneous operations, but you do NOT need to implement it this way since we are not building a full pipeline with composite operations. Also, your clipping operations are creating and using outcodes, but you do NOT need to perform the

clipping operation since there is already line clipping code in src/clip.cpp. Finally, you may want to call some shading functions inside other shading functions. This is why they were created in the first place.

3. Implementation:

For this project, you do NOT need to create an OpenGL program that displays anything. All you need is a very simple main program that calls all of the functions above with a variety of parameters to demonstrate that they are all working. Using random number generators will help you stress test all of your functions.

Remember to use good programming style when creating your program. Choose good names for variables and constants, use proper indenting for loops and conditionals, and include clear comments in your code. Also, be sure to save backup copies of your program somewhere safe. Otherwise, you may end up retyping your whole program if something goes wrong.

4. Testing:

Test your program with a variety user inputs and/or randomly generated values. You should copy/paste the inputs/outputs of your program into a text file to be included with your code and project report. Please make it clear in your output which functions are being tested.

5. Documentation:

When you have completed your C++ program, write a short report using the project report template describing what the objectives were, what you did, and the status of the program. Finally, describe any known problems and/or your ideas on how to improve your program. Save this report to be submitted electronically via Blackboard.

6. Project Submission:

In this class, we will be using electronic project submission to make sure that all students hand their programming projects and labs on time, and to perform automatic plagiarism analysis of all programs that are submitted. Please upload your project as follows:

1) Please copy your code over to turing, and create a small Makefile to compile your program. Something like this will do the trick (remember to use tab to indent the compile line):

```
# define g++ flags
CC = g++ -Wall -O3

project: project.cpp
	$(CC) -o project project.cpp
```

2) When you get ready to upload into blackboard, please create a tar/zip file that contains your code, your makefile and your report. This way we can download and untar/unzip your code, type "make" and then test it.

The dates on your electronic submission will be used to verify that you met the due date above. All late projects will receive reduced credit:

- 10% off if less than 1 day late,
- 20% off if less than 2 days late,
- 30% off if less than 3 days late,
- no credit if more than 3 days late.

You will receive partial credit for all programs that compile even if they do not meet all program requirements, so handing projects in on time is highly recommended.

7. Academic Honesty Statement:

Students are expected to submit their own work on all programming projects, unless group projects have been explicitly assigned. Students are NOT allowed to distribute code to each other, or copy code from another individual or website. Students ARE allowed to use any materials on the class website, or in the textbook, or ask the instructor and/or GTAs for assistance.

This course will be using highly effective program comparison software to calculate the similarity of all programs to each other, and to homework assignments from previous semesters. Please do not be tempted to plagiarize from another student.

Violations of the policies above will be reported to the Provost's office and may result in a ZERO on the programming project, an F in the class, or suspension from the university, depending on the severity of the violation and any history of prior violations.