CSCE 4813 – Programming Project 2
Due Date – 03/01/2024 at 11:59pm

1. Problem Statement:

The goal of this programming project is to develop a graphics program that displays an object flying along a user generated path on the screen. There are three parts to this project.

Part 1: First, you need to create a polygon model for the object you wish to fly across the screen. You can use any combination of solid shapes (rectangles, sphere, cones, surfaces of revolution) that you wish. Once you have this model defined write an OpenGL program that displays your object centered on the display window.

Part 2: In order to define the flight path, you need to use the mouse callback function to capture a sequence of (x,y) coordinates as the user draws the path. You may wish to use mouse-click-down to start the flight path and mouse-click-up to end the path. You will need to convert the (x,y) values from screen coordinates to object coordinates at some time.

Part 3: Extend your program to produce an animation of your object flying along the user-defined path. Try to time your animation so the object flight takes a few seconds so we can admire your object flying across the screen. You are welcome to use any OpenGL callbacks you wish to control this animation.

If you want to get really fancy, you can rotate your flying object so the front of the object is always pointing in the direction of motion like the UFO in the figure above.

2. Design:

Students can design any polygon model they wish for this project. You may want to look at the sample code for creating office furniture to see how functions can be used to create objects with multiple parts (like bookcases, tables). There are also several programs that create objects of revolution you can look at.

Mouse callback functions will be needed to capture and store the (x,y) coordinates of the intended flight path. Be careful to keep track of how many points you are storing, or you may have an accidental array bounds error.

In order to animate the motion of your flying object you will need to create a transformation matrix with the operations in the correct order. It will be easy to see when you have a bug in your code because your object will fly in the wrong direction or rotate incorrectly. Remember, the first operation you apply to the transformation matrix will be the last operation applied to your object.

3. Implementation:

This semester we will be using C++ and OpenGL to implement all of our programming projects. The instructions for downloading and installing a Linux VM and installing OpenGL are posted in README file the "Source Code" page of the class website. Once you have OpenGL installed, you can compile your graphics program using "g++ -Wall animation.cpp -o firework -lGL -lGLU -lglut".

You are encouraged to look at sample OpenGL programs to see how the "main" function and the "display" function are normally implemented. As always, you should break the code into appropriate functions, and then add code incrementally writing comments, adding code, compiling, debugging, a little bit at a time.

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 different random number generator seeds until you get some images that look fun/interesting. Take a screen shot of these images to include in your project report. You may also want to show some bad/ugly images that illustrate what happens if there is a problem somewhere (e.g. too many lines, lines too short, etc.). You can discuss how you corrected these problems in your project report.

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. Be sure to include several output images. 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. When you have completed the tasks above go to Blackboard to upload your documentation (a single docx or pdf file), and all of your C++ program files. Do NOT upload an executable version of your program.

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.