ECE 417 / 617 Individual Project

In this project you will create a consumer-level drawing application. Your program should allow the user to draw multiple polylines and polygons. At any time, the user should be able to change the current drawing color, line thickness, and fill color (for polygons). The user should also be allowed to change these values for existing polylines/polygons, and the user should be able to change the drawing order of the polylines/polygons (i.e., which ones are drawn on top in case of overlap). Your program should allow the user to load and save a file (using a file format that you develop). The program should automatically keep track of whether the drawing has changed, and, if so, to prompt the user upon exit.

Your program must compile under Visual Studio 2010, and – apart from code generated automatically by the VS2010 wizard – it must not contain any code written by anyone else without prior approval from the instructor. You should also produce documentation for both the user and the (fictitious) software team responsible for maintaining your code. Your code must be checked into CVS in a subdirectory named with your Clemson user id, in a directory specified by the instructor. You should not check out the subdirectories of any other students.

The project statement is intentionally open-ended to some extent. It is up to you to make wise design decisions to produce a product that an average customer would want. The purpose of the project is not only for you to write code to solve the problem, but also to approach the project with a software development mindset in order to produce code that is clean, readable, maintainable, secure, efficient, and reliable.

Checklist:

- Are the data and display code cleanly separated?
- Are the data cleanly organized into classes?
- Is memory managed efficiently? Are there any memory leaks?
- Does the program ever crash?
- Does the program work in both Debug and Release modes? Can the developer step through the code in Debug mode to demonstrate the program's functionality?
- Is the code well commented?
- Is the software license for the code clearly documented at the top of each file?
- Are spaces used appropriately? Are coding conventions followed, or reasonable alternatives?
- Is the code easy to read?
- Is the file format easily extensible?
- Is it easy to read and edit files outside of the application?
- Does the documentation clearly explain how to interpret the data structures?
- Does the documentation clearly explain how the code is organized?
- Does the documentation clearly explain the rationale behind any important decisions made by the developer?
- Does the documentation clearly explain how to use the program?
- Is the documentation neat, well organized, and free from typos and grammatical errors?
- Was CVS used properly?

Grading will be done holistically. Three scores are possible:

- A project with a score of 3 will have all the functionality implemented. The program will function as expected by the user; and the code will be cleanly written, well organized, clearly commented, and easy to follow. The program will never crash. Coding conventions will be followed, and appropriate variable names will be used based upon the scope of the code. Documentation will be professionally written. CVS will be used properly, with all of the necessary files (and no unnecessary ones) checked in.
- A project with a score of 2 will have nearly all the functionality implemented. Some minor bugs in the program may be visible to the user, or some minor aspects of the program may be incomplete or not work properly. Code will be well organized, but some parts may need explanation by the developer to follow the code. Documentation will be reasonably clear but lack final polishing.
- A project with a score of 1 may have only the basic functionality implemented. Some functionality may not be implemented or may not work properly. The program may crash upon occasion. Code may be poorly organized, difficult to follow, or not well commented. Documentation may be sloppy or unclear.

A score of 0 will be assigned to a project that is not turned in, or for which the student has obviously not made a serious attempt to solve the problem.

The project is due on the date posted on the website. The instructor/grader will perform a CVS update shortly after the deadline and assign a score based on the files checked in at that time. Be sure that all code necessary to compile and run the program, as well as all documentation, has been checked in properly. At the request of the student, a new CVS update will be performed at an earlier or later date, and the grade will be raised or lowered accordingly. For this to happen, the student must send an email to the instructor/grader with a subject line of "ECE417/617 grading request". The timestamp on the email will be used for the CVS update.

The grade will be assigned as follows. Let d be the number of days late that the assignment is graded (or, equivalently, $-d$ be the number of days early). These days are counted using a 6-day work week. The grade (out of 30), is then $10 * score + r$, where

- $r = 6$ if $d \leq -7$ (reward for turning in more than one week early)
- $r = 0$ if $-6 \leq d \leq 0$
- $r = -d$ if $1 \leq d \leq 6$ (penalty for turning in up to one week late)
- $r = 6*ceiling(d/6)$ otherwise. (penalty for turning in more than one week late)

A second grading will occur shortly after the second deadline in the same manner. This will give students an opportunity to improve their grade based on feedback on the first submission. The final grade will be an average of the two grades. If no changes are made, then the first grade will be the final grade. For the second grading to occur, the student must send an email with a subject line of "ECE417/617 grading request #2".