Introduction to OpenGL Programming (part-2)

Objective:

1. Becoming familiar with OpenGL programming structure using OpenGL Utility Toolkit (GLUT).
2. Learning how to initialize windows and objects inside a graphics window

Reference:

1. OpenGL Programming Guide, Red Book, Chapter-1

Prerequisite:

Knowledge of:

- C/C++ Programming
- Basic Coordinate Geometry

Academic Honesty:

All work that you do toward fulfillment of this course's expectations must be your own unless collaboration is explicitly allowed (e.g., by some problem set or the final project). Viewing or copying another individual's work (even if left by a printer, stored in an executable directory, or accidentally shared in the course's virtual classroom) or lifting material from a book, magazine, website, or other source—even in part—and presenting it as your own constitutes academic dishonesty, as does showing or giving your work, even in part, to another student.

Similarly is dual submission academic dishonesty: you may not submit the same or similar work to this course that you have submitted or will submit to another. Nor may you provide or make available your or other students' solutions to individuals who take or may take this course in the future.

You are welcome to discuss the course's material with others in order to better understand it. You may even discuss problem sets with classmates, but you may not share code. You may also turn to the Web for instruction beyond the course's lectures and sections, for references, and for solutions to technical difficulties, but not for outright solutions to problems on projects. However, failure to cite (as with comments) the origin of any code or technique that you do discover outside of the course's lectures and sections (even while respecting these constraints) and then integrate into your own work may be considered academic dishonesty.

All forms of academic dishonesty are dealt with harshly.

Evaluation Policy:

Your code will be evaluated along the following axes.

Correctness. To what extent is your code consistent with our specifications and free of bugs?

Design. To what extent is your code written well (i.e., clearly, efficiently, elegantly, and/or logically)?

Style. To what extent is your code readable (commented and indented with variables aptly named)?

Attachment: