Lab Requirements
CSCI 4250/5250

Language and platform

Labs are to be written using Visual Studio 2008, OpenGL, and GLUT. You will turn in a zipped project folder (without the debug folder). Turn in your project to Desire2Learn (D2L) at http://elearn.mtsu.edu. In D2L, click on dropbox, click on the appropriate assignment, and upload your zipped folder. You may do this multiple times. I will only grade the last version of the project turned in. To grade your project, I will unzip your project, rebuild it, and run it on my machine.

If I have any trouble compiling or executing your program, I will notify you by e-mail (your MTSU e-mail address will be used) as soon as possible. Thus read your e-mail.

I will also obtain a printout of your source code using Visual Studio. The font will be set to Courier size 10. Avoid wraparound (i.e. make your lines short enough that they will fit on one line of an 8 ½ by 11 page using Portrait setting for the printer.

Due Date

Late labs will be docked 10% per class day. Labs will be counted on time if they have a date on or before 11:59 pm on the due date. Labs will not be accepted after graded labs have been returned.

Correctness (at least 50% of the total lab grade but possible up to 70%)

When I receive your lab, I will compile it and run it. Programs that produce syntax errors will not be graded. When syntax errors occur, I will inform you immediately via e-mail in hopes that problems occurred in transmission. You should make sure that your program works on computers in the lab before turning it in to me.

Most graphics labs do much more than produce a picture. They are usually interactive in nature and the user is allowed to make a variety of choices. When I grade your lab, I test the program thoroughly to make sure that it meets the specs and produces the correct images. You should thoroughly test your program before turning it in. Partial credit will be given for a partially working program.

If I am completely unsuccessful running a program, I may require a meeting with you so that you can demonstrate how to use your program.

Documentation (up to 20% of the lab grade)

Your application program (the file containing the main function) should include a heading at the top of the program with the following information:

FILE NAME:	main.cc (or whatever the name of this file is)
PROGRAMMER:	Your name
CLASS:	CSCI 4250/5250
DUE DATE:	date in the form Monday, 8/19/08
INSTRUCTOR:	Dr. Hankins

The heading should also include the following in **complete correct sentences**

- a short description of the program
- a description of user input if any
- a description of expected results
- tips for using the program
- limitations – how much of the program is working and what limitations do you perceive in the program

In addition to this beginning documentation, each section of the program should be fully documented. A heading that contains the following should precede each function or method:

- the name of the function/method to follow
- the purpose of the function

All variables **must** be briefly described (this includes local variables as well as function parameters). Comments beside variables should be aligned as much as possible in a column.

When using a separate class, the header (.h) file should contain beginning comments giving the file name, the name of the class, the purpose of the class. Brief documentation should precede each function prototype and data member in the class. The implementation file (.cc) file should contain a duplicate of the beginning information placed in the .h file. In addition, each method is a function and should contain function headings as described above.

Enough comments should be spread throughout the program so that the reader can easily follow the logic.

Maintainability (up to 30%)

- Structured programming techniques should be used. Each function should have a single well-defined purpose.
- Use proper indentation and be consistent with your indentation
- Blank lines should be used for readability
- Use descriptive variable names
- Global variables are normally strongly discouraged. However, when using OpenGL and GLUT, it may be necessary to use some global variables. When it is absolutely necessary, you will can use global variables. However, avoid them when possible!

Efficiency (up to 20%)

Care should be exercised to select efficient algorithms that do not have excessive space and time requirements.