Assignment 1 - Programming OpenGL

(Note that there is 55% of the grade in programming projects. This assignment will be weighted to be 5 of those 55 points, so 5% of your total class grade)

Notes for CSE 470/598
Instructor: Ross Maciejewski

1 Project Guidelines

- Have fun, learn, be creative, and create something you are proud to show-off!
- No teamwork allowed! It is ok to discuss the projects with other students. You can use the discussion board on myASU (The TA will also try to answer your questions posted at myASU, but only sporadically); however your code must be your own. (You would be surprised how easy it is to detect copied code!). You cannot post code on the discussion board! (only very small code sections)
- Comment your code - you will be glad you did. You will be reusing code throughout the semester.
- You will upload your projects on myASU.
- If you have **good reasons** that you cannot complete a project on time **and** you have written documentation, then we can make adjustments to due dates. However, you must notify us before the due date that you would like to discuss such an arrangement. Good reasons would be illness, family emergency, visiting a conference to present a paper, …

2 Getting Started

The projects assume that you are using: Microsoft Windows, Visual C++, and OpenGL

2.1 Downloading the Compiler

2.1.1 At Home

For your home computer you can download the compiler from:
http://www.microsoft.com/downloads/
Search for: **Visual C++ Toolkit**
Install the toolkit.
If you are a student at CSE, you may get the free Visual Studio from ASU.

2.1.2 In the lab

In the lab the compiler should be installed.

2.2 Getting your programming environment ready

2.2.1 Visual Studio .Net

- Start Visual Studio .NET * File → New → Project * Select Visual C++ Projects → Win32 → Win32 Console Project
    - Give your project a name and choose where to store it
    - Click Finish

- Disable support for precompiled headers
    - Remove the '#include "stdafx.h"' line from your project1.cpp
    - Open Project -> projectname Properties (last option)
    - Go to C++ -> Precompiled Headers
    - Change "Create/Use Precompiled Header" field to "Not using precompiled headers"
- Add glut32.lib to your linking settings
    - You should still be in the project settings window
    - Choose Linker -> Input
    - Add 'glut32.lib' to your Additional Dependencies
    - Click OK to return to your main project window

Now we need to load the glut files

- Put the glut32.dll in the same directory as your executable
    - Download from myASU (glut-3.7.6-bin.zip 117 KB). Note: glut is also freely available on many internet sites.
    - When the zip folder opens, copy “glut32.lib”, “glut32.dll”, and “glut.h” to the directory you stored your project in
    - Now, any time you see '#include <gl/glut.h>' replace it with '#include "glut.h"'
- You should now be able to build your project
- Also, if you paste in code from any of the examples it should work

2.2.2 Visual C++ 6.0 (outdated)

The GLUT files go into folders that are mirrored by their OpenGL entries. I would recommend searching for the opengl files mentioned below to find the folders specific to your machine.

1. Your "WINDOWS/system32" folder should already contain “openGL32.dll” and “glu32.dll”. Copy the “glut32.dll” here.
2. The “opengl32.lib” will live in a folder named something like "..\Microsoft Visual Studio\VC98\lib". Copy the “glut32.lib” file here.
3. Similarly, the “gl.h” file will live in a folder like "..Microsoft Visual Studio\VC98\include\GL". Copy “glut.h” here. (Note that “glut.h” defines all the necessary opengl include files, so you only need to include it in your program, instead of including “gl.h” and “glu.h” as well as “glut.h” together)

To start your own program in VC++, do the following.

1. Start VC++
2. File->New->Open a console application
3. Select an "empty project" and pick a name and directory
4. File->New->C++ source (pick a name and directory)
5. You are ready to go! (See a sample program for the basic includes and program structure.)

If you are not allowed to put files in the system and VC folders, then do the following.

1. Put the GLUT files in a folder of your choosing.
2. Open VC++
3. Choose Tools->Options
4. Choose the "Directories" tab.
5. In the *lib, include,* and *executable* sections, add the folder in which the GLUT files live.
6. With this set-up, the <GL/glut.h> in your program needs to be changed <glut.h>

2.3 Download and Compile two sample programs

To test your installation a first good step is to download and install two sample programs from myASU.

3 C and C++

C and C++ are closely related to Java. If you had no previous exposure to either C or C++ you should try reading a tutorial on the internet. You can search for "C for Java programmers" or "C++ for Java programmers" on google.
For OpenGL you will start out using only the C part and then later on use some C++ functionality to structure a more complex program.

Example:
C for Java programmers -
http://www.cs.cornell.edu/courses/cs414/2001SP/tutorials/cforjava.htm

C++ for Java programmers –
http://www.cs.brown.edu/courses/cs123/javatoc.shtml

4 Upload your projects

How to upload a Project
This is a short guide for the successful upload of a project:

- Check the due date of the project on myASU
- Use myASU (digital dropbox) to upload your project. (Do not send the program as email attachment to the instructor or TA!)
- Your project should be packed in a zip file and named with the following naming convention:
  PX.Lastname.V1.zip
  (if you submit some updated files, use a different version)
  So my second project would be named
  P2.Maciejewski.V1.zip
  If I then found a last minute mistake and want to upload again I would use the name P2.Maciejewski.V2.zip

- If there is someone with the same last name in the course you can add your first name after the last name in the filename.
- The zip file should include all the source code and should be ready to compile. That means it should include your Visual C++ project files.
- The zip file should include an executable and the glut32.dll so that it can be started directly by clicking on it. (from wherever the file is downloaded to).
- A short text file called "instructions.txt" – this text file should include the following:
    - a) instructions about how to navigate in your program, including the function of special keys (e.g. x exits the program)
    - b) special functions that you implemented
    - c) changes from the original specification that you negotiated with the instructor or TA to the original specification
    - d) other things you consider important
    - e) specify the libraries or code that you used and did not write yourself. You do not need to specify code that we provide on myASU.
- You can use rar instead of zip to compress your files

5 Project 1

5.1 Purpose

This project has been designed for an intro to OpenGL. You will learn how to draw simple shapes on the screen and add menu functions and mouse interaction.

5.2 Overview

You should take the sample code provided on blackboard and compile the code.

5.3 Required Functionality

Here are the elements that your program must have. Despite this "laundry list" of requirements, there is some room for creativity! Note: If the text suggests a menu item "Sound - Off" that would mean that "Off" is an entry in the submenu "Sound".

- **(10p) Create a drawCircle Function** – Fill in the draw circle function in the program so that if called it can create 360 vertices for a circle to be drawn on the screen.

- **(5p) Draw on the index card** – You will create a menu entry "Draw" with subentries "square" and "circle" that when clicked will draw either a square or a circle in the center of the index card on the picture.

- **(5p) Debug helper** – You will create a menu entry "Images" with subentries "1" and "test" that when clicked on will change the background image to either image 1.png or test.png (note the images will be flipped and that's ok for this part for undergraduates, graduate students see your extra part)

- **(20p) Output the color of a pixel on the screen** – Fill in the myMouse function such that when a user clicks on the screen, it outputs the corresponding color in a printf statement to the background console window saying "Screen click, R = xxx, G = xxx, B = xxx." Where the xxx are the actual values from the *image array with values from 0 to 255. (Hint: The *image is an unsigned character array. This is just the image colors for each pixel R,G,B laid out in a giant linear order. You need to figure out how to index into the array intelligently.) (Hint 2: To properly print an unsigned char, you would not use %c, you use something else.)

- **(10p) Graduate Student Only Requirement –** You will have noticed that the image drawn on the screen is inverted. Write a function that will reorder the input array such that the output on the screen is no longer inverted. As such the image shown on your screen should be shown in the correct position and you will then need to update other items to draw things correctly in the center of the index card.

5.4 Grading

Note that there are 40 points available to undergraduates, 50 to graduate students. All assignments will be graded out of 50 where undergrads get a scaled grade, if they get 40/40, then their final score will show as 50/50. Undergraduates are allowed to complete the graduate portion for 5% extra credit on the assignment meaning that an undergraduate that completed all of the work on this assignment would receive a 52.5/50.

These points represent an initial weighting of the functionality. **Important:** You get full points for a *good* implementation of the required functionality. In principle you will get full points if you implement the requirements correctly, but there are many ways to do things according to specification that we might consider *unreasonable* or *undesired*. Especially we will deduct points if the design choices make it difficult to judge the program (E.g. you choose materials that are so dark that it hard to see anything, the objects move too slowly or too fast, the control of the program is extremely difficult.)

Note: we might change the weights slightly if a requirement is misunderstood by a majority of the class or some other special circumstance. I do not expect that to happen though.