COMP 308
Software Systems Lab

Assignment #3

Due: Friday April 6th, 2018 on myCourses at 23:30

This assignment has two parts: Assembler graphics programming and Higher-level graphics programming (OpenGL and C). It also expands on assignment #2's graphics library.

PART 1: Graphics Programming and Assembler

From assignment #1 and #2 you now have a good IO.ASM "library" source file and we just began experimenting with the GRAPHICS.ASM library in ass#2. In this assignment we will complete a graphics "library" source file called GAPHICS.ASM. You will need to be able to use what you saw from ass#2 on retrieving graphic card information.

Creating your own Graphics Library

Basic instructions:

- For this assignment you can use VESA BIOS calls except for drawing a pixel. You must access the memory buffer directly for pixel drawing (as we did in class)

Assignment instructions:

This assignment will combine many of the things you have seen in class and the things you have learned from Assignments #1 and #2. This assignment asks you to create a basic graphics API. You will also create a main program that will use the API you created to validate it.

Create the file GAPHICS.ASM and place your API within this file. Then create a main program file called MAIN.ASM that will invoke your API. You are permitted to merge these files into a single file (together with IO.ASM) called ASS3.ASM for compilation and execution.

GAPHICS.ASM:

The GAPHICS.ASM API provides basic graphics API functions. Please provide the following functionality:

- bool AL SETMODE(int AH)
  - Permits the user to set the computer to some mode. If the mode does not work auto reset back to the standard text mode, then return with false.

- bool AL SETPENCOLOR(int COLOR)
  - In order to draw you must first select the pencil you want to draw with. This function does this for you. By providing the color number as a parameter the API will remember that you are currently using that colored pencil. The function returns false if it encounters a problem. You will need a variable in memory to store this information, call it PENCOLOR.

- bool AL DRAWPIXEL(int X, int Y)
  - Using the pen color and mode you already selected set pixel at X, Y to color and returns false if there was a problem. **Important**: this function does not use the BIOS but accesses the memory directly. This is from ass#2.

- Void GET_CARDINFO (void)
  - This will query the graphics card for it's information and save it in a buffer. This is similar to what you did in assignment #2. The other library instructions may need to use the downloaded information.

- bool AL DRAWLINE(int X, int Y, int X2, int Y2)
  - Using the pen color and the mode you already selected this function draws a **general** line from the two coordinates in any direction. This is a general purpose line drawing function. It returns false if there was a problem. Note that this function uses DRAWPIXEL to actually write on the screen buffer.

- Void SIMPLEFILL(int x, int y)
  - This function assumes the color has already been selected and assumes a simple shape. You can use the simple fill algorithm we talked about in class for open (simple) shapes. Use DRAWPIXEL to actually write on the screen buffer.

Your Main program: ask the user to input a mode number and two color numbers, then switch to that mode and draw a single triangle using the selected first color and fill it with the selected second color.

PART 2: OpenGL

To help you out, I've uploaded to the “Lecture and Handouts” section of My Courses the sample program cube.zip. The cube.zip file is a simple Open GL example. This question is based on this sample program. I think getting the sample source code to run will be the hardest part of the assignment. Once it is running, I think the rest will be easier than our previous assignments. (Tell me otherwise). Note that Lab 7 is also very helpful for this assignment (also already posted on My Courses)

The TA will have a lab to help you install and get running the above program.

To help you install it on your own, follow the following instructions:

- For the glaux library, download the Microsoft Platform SDK (also uploaded to My Courses).
- To make the project compile properly, you should add in Visual C++ under Project->Properties->(open window)->Configuration Properties->C/C++->Additional Include Directories, you should add here:"C:\Program Files\Microsoft Platform SDK\Include" and "C:\Program Files\NVIDIA Corporation\Cg\include".
- and under Project->Properties->(open window)->Configuration Properties->Linker->Additional Library Directories, add "C:\Program Files\Microsoft Platform SDK\Lib" and "C:\Program Files\NVIDIA Corporation\Cg\lib".
- This should get the test project to compile and link, provided you've installed the Microsoft Platform SDK and the Nvidia Cg Toolkit. Note that you need to also provide a bitmap image under "/data/image.bmp" directory of the project for it to load this texture on the cube it will display.

Home Scene using OpenGL

Using OpenGL construct the following scene: Create a simple stick figure scene of a person walking to their house in the foreground. The background will be blue (for sky), green (for grass) and a distant sun as a small yellow filled circle (with seven yellow rays). You can position these elements and format their shapes in any way you like. Be creative. No moving parts.

Use cube.zip and the lecture slides to help you construct the solution.

Optional: implement the windows interface using the simple technique shown from this web site:
See: http://www.swiftless.com/opengltuts.html.

HOW TO HAND IT IN

Submit GAPI.ASM, ASS3.ASM, A3Q1.C, any image files, and any other .cg files you used, ZIPPED to My Courses. Also include the compiled version of your programs. Make sure this is executable in the Trottier Windows lab.

HOW IT WILL BE GRADED

This assignment is worth a total of 20 points

- GAPHICS.ASM
  - +6 (one point for each function)
- GAPHICS Main Program and getting it to run
  - +4 points
- OPEN GL:
  - Foreground +2.5
  - Background +2.5
  - Getting OpenGL to run +5
- Optional: implement windows as in swiftless.com's recommendation

GRADING RULES

- Your program must run to be graded, or you will get zero.
  - If you program runs but does not run correctly, you will get part marks
  - Part marks are awarded proportionally, meaning that it is only based on the parts of the program that you actually did complete correctly.
- Essay / Written questions are also graded proportionally. This is defined as points being awarded to only those portions of the question that were answered correctly.
  - If you answered, for example, only half of the questions and they were all correct then you get half of the points.
- Late submission penalty : 10%/day