

CS175 - Assignment 1

Hello World (OpenGL)

Due:

Monday, September 16th, 11:59pm

1 Objectives

The purpose of this assignment is to give you experience with programming simple OpenGL applications. You will also get some initial experience with programmable shaders.

You can download the assignment 1 starter code (C++) at: <http://www.fas.harvard.edu/~lib175/asst/asst1.zip>

The following is a brief description of the files you will find:

- `HelloWorld2D.cpp` contains the `main` function and has code to
- `ppm.[cpp,h]`: provides functions for reading/writing images in a very simple image file format (PPM).
- `Makefile, Makefile.mac`: makefiles for Linux and Mac users.
- `reachup.ppm, mountain.ppm`: example image files (in PPM format) to experiment with.
- `HelloWorld.[vproj,sln]`: solution and project files for VisualC++.
- `README.txt`: more detailed information about files.

You should write your own `README.txt` file for assignment submission.

1.1 PPM Image format

Feel free to create your own PPM texture files. If you do so, you'll want to note that OpenGL works best with textures that have dimensions that are powers of two (e.g., 256x256, 512x512, ...). On the Science Center linux and Mac machines, `xv`, `gimp`, `convert`, and `display` may be convenient for image viewing and editing. On Mac and Windows, Photoshop provides a lot of relevant features. Windows users can download a free PPM viewer called `ifranview` to view PPM files. The PPM image format itself is quite simple, but you do not have to understand it to do the assignments: it has a header giving the dimensions of the image (`n, m`) followed by `n*m` triplets of bytes representing red, green, and blue. The pixels in a ppm file appear from left to right within a scanline (**row-major order**), with scanlines appearing from top to bottom. PPM files can be written in ASCII text mode or binary mode. We use the binary mode in our provided functions

for compactness. Therefore, when editing and saving files in your chosen paint program be sure to choose the binary option.

We provide you with source code for functions that read in a ppm file, and store the data into a (n*m) array of pixels. A pixel is represented as three bytes (unsigned chars). The functions we provide flip the vertical order, so that when indexing into the array, we can interpret the coordinate system to be (x,y) with x going left to right, and y going from bottom to top. You can look specifically at the function:

```
packed_pixel_t * ppmread(const char *filename, int *wp, int *hp);
```

for more information.

2 Task 1

The first step is to set up your programming environment so that you can begin to work with OpenGL programs. You will need the following:

- **Hardware support** You will need hardware that supports OpenGL Shader Model 2.0. In practice any computer purchased within the last 2 or 3 years should support it. To test this, you can compile and execute the code provided, if your hardware does not support Shader Model 2.0 then the program will output a message notifying you of it.
- **Driver support** Even if you have a new or up-to-date graphics card installed in your machine, your programs may not be able to utilize the advanced features unless you have installed new drivers. OpenGL is a continually evolving standard and the various graphics chip vendors make regular releases of drivers exposing more and more advanced features (as well as fixing bugs in previous versions). If you are working on your own computer, you will want to download and install the latest drivers. If you are working on the Science Center machines, this has already been done for you. On Intel-Macs this is expected to not be an issue (if it does not work on your Intel-Mac, please email the course staff at lib175@fas.harvard.edu about it).
- **Installing GLUT and GLEW.** GLUT and GLEW are cross-platform libraries that make interfacing with OpenGL and OS specific tasks easier. GLUT handles windowing and message control. GLEW loads the various advanced extensions that have come out since OpenGL's conception. You will need to download and install the headers (.h), and binaries (.lib, .dll/so/a) for them. If you are working on the Science Center machines, this has been done for you. More info can be found on the class website under the Miscellaneous heading at: <http://www.fas.harvard.edu/~lib175/resources.html>

Once you get all the hardware and library issues sorted out, build the program and make sure it runs. If there are remaining outstanding issues, feel free to e-mail the course staff at lib175@fas.harvard.edu.

3 Task 2

Now that the build environment is set up, it is time to get experience playing with OpenGL. Modify the program so that when the window is reshaped, the aspect ratio of the object drawn does not

change (i.e., if a square is rendered, resizing the window should not make it look like a rectangle with different edge lengths), and it doesn't crop the image. Making the window uniformly larger should also make the object uniformly larger. Likewise for uniform shrinking. Figure 1 shows examples of how a window should look when resized:

There are some important constraints:

- Your solution cannot change the vertices' coordinates that are sent by your program to the vertex shader.
- Your solution cannot modify the call to `glViewport` (which is in the `reshape` function in `HelloWorld2D.cpp`).
- **[Hint:]** Your solution can modify the vertex shader, as well as change uniform variables that are used in the vertex shader.

4 Task 3

This is the creative part of the assignment. Browse through the documentation for OpenGL Shading Language (GLSL) and get familiar with some of the built-in operations available. Edit your vertex and/or fragment programs to achieve novel effects. Examples might be using the built-in sine and cosine functions to create undulating patterns. Or perhaps reading in multiple texture files and combining them in some interesting way. Have fun!!!

5 Submission

To submit your assignment, you will first have to log on `ice.harvard.edu` (note: logging into `fas.harvard.edu` will not work). Once in your account, and from the prompt, you will want to change directories into the directory containing the files you want to submit. Inside that directory, type: `submit lib175 1 'pwd'` where the ticks are not apostrophes but back-ticks. Instead of `'pwd'`, you can also write down an absolute path like `/src/asst1`, if that is where your assignment is. The submit script works with absolute paths and will submit all the files and subdirectories within the specified directory. You should get a message claiming success or failure after running the script. (Please do not submit your whole user account: do not execute submit from your home directory). In addition to your source code, you should also include a `README` file. Details on submission can be found at: <http://www.fas.harvard.edu/~lib175/pages/asstinfo.html>

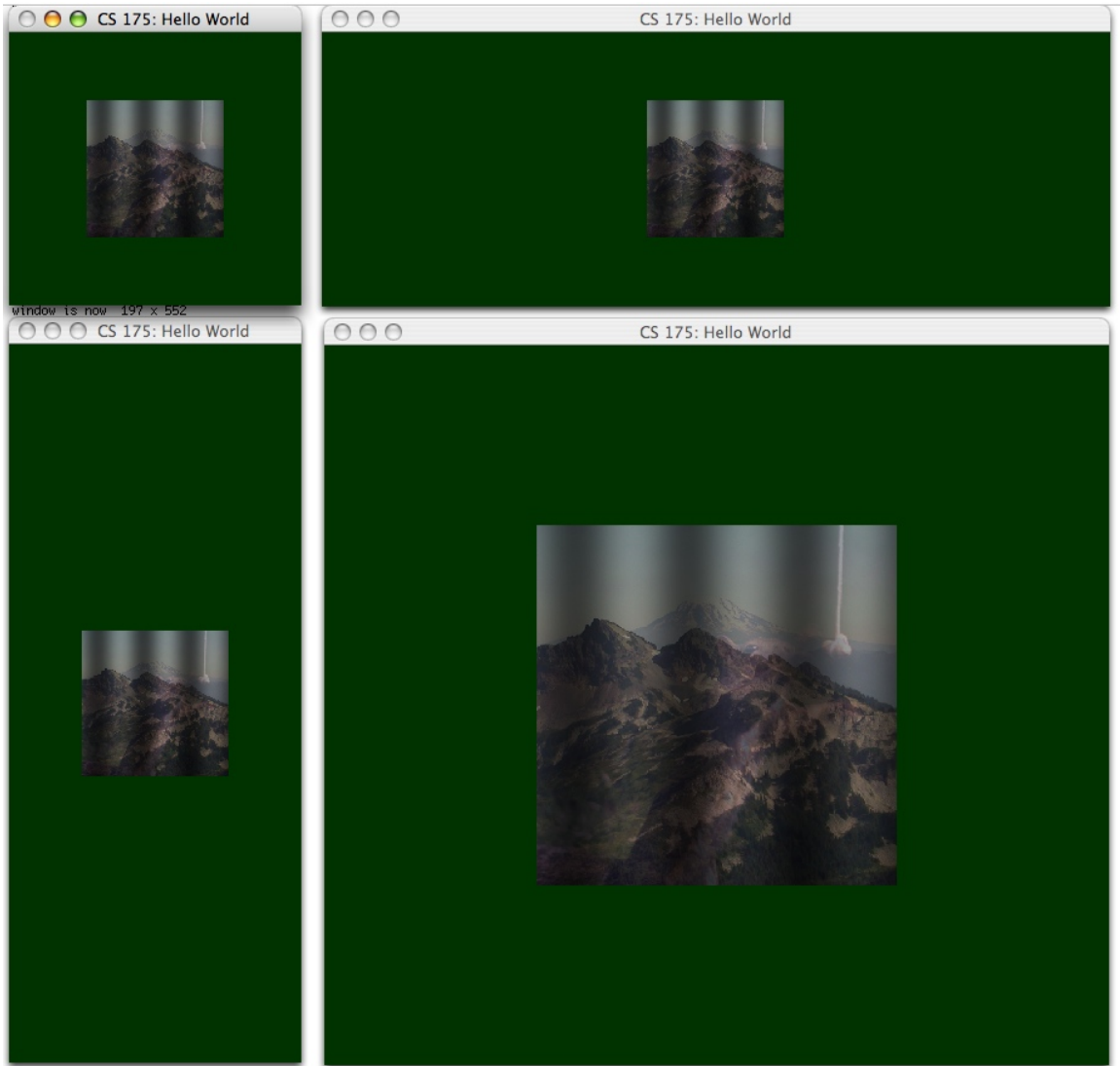


Figure 1: Some sample resized windows and the expected behavior: the drawing is not cropped, and resizing tries to make the drawing as large as possible within the window.