CPSC 4040/6040
Computer Graphics Images

Assigned: Aug 25, 2020
Due: 11:59pm, Sep 5, 2020

Programming Assignment 1 — Get the Picture!

(Grading: 0–10 points)

Problem Description

You are to write a program that reads an image from a file, displays the image, and writes the displayed image to an image file. You will use OpenGL and GLUT for displaying images. In addition, you will be able to handle almost any type of image file by making use of OpenImageIO's API for the reading and writing of images.

Your program should contain the following four distinct procedures: one to read from an image file into your program and convert it to an RGBA pixmap, one to display the image stored in the pixmap in a window sized to exactly fit the image, one to alter the displayed pixmap, and one to write an image file from the RGBA pixmap. This will give you three key building blocks for future assignments, where you will be inputting images, manipulating them in various ways, displaying them, and saving the result.

Basic Requirements

The program should be named `imgview`, and should handle images stored in any format supported by OpenImageIO. The file `cube.ppm` is a simple test image to try your program with. If your program works with this image, try it on the image file `teapot.jpg`, then try it on images of other sizes and formats. In general, your program should be able to display any type of image, from a bitmap and graymap to truecolor images and truecolor with alpha. You should be building off of this code for the rest of the semester, so the goal is to have a generic visualizer.

You should use the OpenGL routine `glDrawPixels()` to draw an entire array of pixel values to the display in one call. Note, that the RBGA format contains an alpha value as well as red, green, and blue values for each pixel. The alpha value is a measure of opacity, and is used for multiple image blending and compositing. For now you can simply set the alpha byte for each pixel to 255, which means fully opaque, or if the image file was in RGBA format you should use the alpha value from the image for each pixel (test, e.g., the `mario.png` file).

The program should use an optional command line parameter to determine the initial input file name. For example, if the command line is

```
imgview cube.ppm
```

the program will first read and display the image file named cube.ppm, before waiting for key presses to do any additional image reads/writes or processing operations. If the program command line is simply

```
imgview
```

then the program should display an empty window of a convenient size, with a black background.

The program's user interface should respond to keyboard key presses as follows:

- 'r' or 'R' key, the program should prompt the user for an input image file name, read the image file, and store it in an appropriately sized pixmap and update the display to show the image from the pixmap.

- ‘w’ or ‘W’ key, the program should prompt the user for an output image file name, and write this image file from the pixmap.
- ’i’ or ’I’, the program will invert the colors of the image. Upon inverting, you should update the displayed image. Unlike with reading/writing, you do not need to involve OpenImageIO for this functionality – you can use your intermediate data structure and write the function so that it updates that data.
- ‘q’, ‘Q’ or ESC key, the program should exit.

Important note: Before you begin coding, you should also choose the data structure that you will use to store the image. This needs not to be complicated (e.g., you could use an array of unsigned char), but you may want to consider employing object-oriented structures (e.g., a class for a pixel or an image) and contiguous memory allocation. We will be using images the entire semester, and you may want to take the opportunity to encapsulate some expected functionality that you believe you will need. The sets of tasks required for this assignment are somewhat representative.

Advanced Extension for 6040 students (extra credits for 4040)

Implement *two* of the following features:

- Allow your program to read multiple images. Using a single display window, allow the user cycle between them using the left and right arrow keys. Do this by specifying multiple filenames at the command line. For example, the command line

  ```
  imgview cube.ppm teapot.jpg
  ```

  will read both the image file named `cube.ppm` as well as `teapot.jpg`. `cube.ppm` will be displayed first. When the user presses the ‘w’ key, whichever image is currently displayed will be written to the specified output file.
- Implement some additional display controls. The first will allow the user to press the ‘1’, ‘2’, or ‘3’ keys and display only the red, green, or blue channel, respectively, as a single channel, greyscale image. Pressing the ‘o’ key should revert the displayed image back to its original form. When the user hits the ‘w’ key, it should write the active window display to the specified. Note that you need to play with the third argument of the `glDrawPixels()` routine, as well as the number of channels in the `ImageSpec` class.
- Implement additional manipulation controls. For example, pressing the ‘v’ key will allow the user to flip the image vertically, and pressing the ‘h’ key will flip the image horizontally. Upon flipping, you should update the displayed image. Note that later in the semester we will learn efficient ways to perform these operations.
- Provide a reshape callback routine for your program, that responds to the user resizing the display window. If the user increases the size of the display window so that it is bigger than the image, the image should remain centered in the window (note that the OpenGL `glViewport()` command will be helpful for this). If the user decreases the size of the display window so that it is smaller than the image the image should be uniformly scaled down to the largest size that will still fit the window (note that the OpenGL command `glPixelZoom()` will come in handy for this).

Code Documentation

1. Include a header comment in each file of your code that gives a description of the program, your name, and any other relevant information (see the online code samples).

2. For each function you create, you should include a brief comment listing what the function does, and what data it produces.

3. Additional inline comments to clarify any complicated things are welcomed.

In addition, your submission should include a README file that includes: (1) your name and email; (2) a brief description of what the code does; (3) and how to run the code (parameters, keyboard or mouse controls, etc.); and (4) any known problems.

Resources

The lectures provided sample code and a brief tutorial on OIIO. On the project wensite, you can also find a sample makefile that we have created. You will need to be well versed in the `ImageSpec`, `TypeDesc`, `ImageInput`, and `ImageOutput` objects for OIIO to complete this assignment. Additionally, you should refer to the OIIO reference manual `https://sites.google.com/a/g.clemson.edu/cpsc-cgi/resources/openimageio.pdf`.

We have also provided a few samples of OpenGL/GLUT code that you can find on the website under Schedule. You will need to understand how to process mouse and keyboard events in a display loop, and how to draw an image on the screen. Chapter 8 of the "red" book is the most relevant, `http://www.glprogramming.com/red/`, along with Glut's documentation that can be found at: `https://www.opengl.org/resources/libraries/glut/spec3/spec3.html`.

Submission

Please write the program in C or C++, using OpenGL/GLUT and OpenImageIO. Please take extra care to make sure that your homework compiles on the SoC Linux machines. Since we are using multiple files, please make a directory named with your Clemson username (lower case), and add all files to it. You should zip your directory and upload it to the Canvas submission system.

Help

If you get stuck, please do not hesitate to contact us for help, and stop by during office hours. We also encourage you to post questions and initiate discussions on Canvas. Your colleagues are also there to help you.