

Lab 06: Image Scaling

Overview

This lab is designed to give students practice with memory manipulation and function as a primer for Project 3. You will be using CLion, along with Windows Terminal and the test images provided on Canvas, to complete this lab. It will make use of a C++ version of the ConsoleGfx class. You will also practice directly executing programs from the command line. (Please note that the CLion terminal does not natively recognize UTF-8 encoding, and the C++ runtime does not do “unnecessary” re-encoding, so proper visualization requires using Windows Terminal and a modified character set. As a result, while this activity can be done on Linux or MacOS, the visualizations will not match the desired result on these platforms.)

Requirements

The program should be driven classless `main` function and `Image` class.


Standalone Functions (scaler.cpp)

`int main()`

When the program is run via the `main()` method, the program should:

- 1) Fetch `ConsoleGfx` singleton via `ConsoleGfx.getInstance()`
- 2) Display the welcome message
- 3) Display color test (`ConsoleGfx.testRainbow`)
- 4) Display the menu
- 5) Prompt for input

```
Welcome to the Image Scaler!

Displaying Spectrum Image:


Scaler Menu
-----
0. Exit
1. Load File
2. Load Test Image
3. Display Image
4. Enlarge Image
5. Shrink Image
6. Show Image Properties

Select a Menu Option: |
```

Load File

Load file via `unsigned char *ConsoleGfx.loadFile(string file)`. Note that `loadFile()` allocates an array for the file data, but the caller is responsible to deallocation.

Success (File was Loaded)

```
Select a Menu Option: 1
Enter name of file to load: logos/uga.gfx
File loaded.
```

```
Scaler Menu
-----
```

Failure (File wasn't Loaded)

```
Select a Menu Option: 1
Enter name of file to load: foo.png
Error: could not load file.
```

```
Scaler Menu
-----
```

Load Test Image

Loads `ConsoleGfx.testImage`:

```
Select a Menu Option: 2
Test image data loaded.
```

```
Scaler Menu
-----
```

Show Image Properties

Displays the length and width of the current image.

```
Select a Menu Option: 6
Image Dimensions: (14, 6)
```

```
Scaler Menu
-----
```

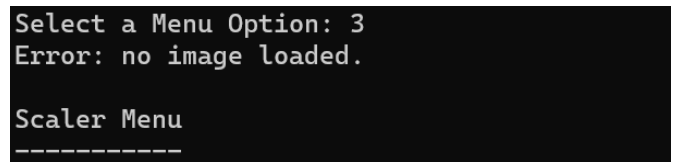
Display Image

Displays current image by invoking `ConsoleGfx.displayImage(unsigned char *imageData)` method.

Success (Previously Loaded)

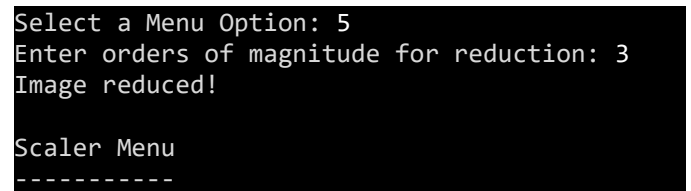
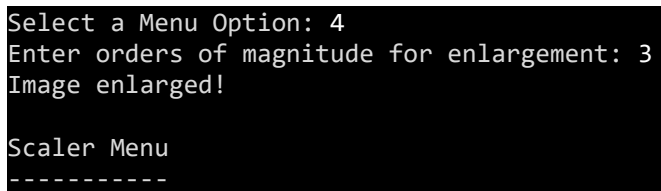


Failure (No Image Loaded)



Enlarge / Shrink Image

Prompts the user for orders of magnitude to change (powers of two), then enlarges the image.



`unsigned char *scaledImage(unsigned char *imageData, int orders)`

Returns a scaled version of the image data based on the **orders** of magnitude (i.e., powers of two to scale). The number of orders of magnitude should be in the range [-4, 4] (inclusive), which allows scaling down to 1/16 or up by a factor of 16 in each dimension. ***Do no implement this via recursive or iterative doubling***; this takes twice as long as creating a new image one time, and it causes memory fragmentation.

This function should never scale any image above 256 in width or height; instead, it should limit scaling such that the image is scaled up by a power of two but remains less than or equal to 256 in each dimension. Likewise, no image should be scaled below a height or width of one. To determine the color for reduced images, the number of colors of each pixel should be counted, and the most common color should be used. If there is a tie, the earliest pixel (by row, then by column) should be used. This function should allocate memory and return it to the caller; the caller is responsible for deallocation.

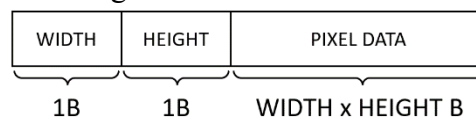
Image Class

The `Image` class will the image data and should be constructed from the raw data.

The class must also have the following methods and behaviors (this is mandatory):

`public Image(unsigned char *imageData)`

This method should be the **only** constructor for this class. There should not be a default constructor! Its sole argument is a pointer to an array of raw image data as loaded from the file. The data is formatted as follows:



`public unsigned char *getImageData()`

Returns a pointer to the raw image data (**not a copy**).

`public unsigned char *getPixelData()`

Returns a pointer to the raw pixel data (excluding the image property information). ***Should not be a copy.***

```
public unsigned char getWidth()
```

Returns the width of the image.

```
public unsigned char getHeight()
```

Returns the height of the image.

```
public void setImageData(unsigned char *newData)
```

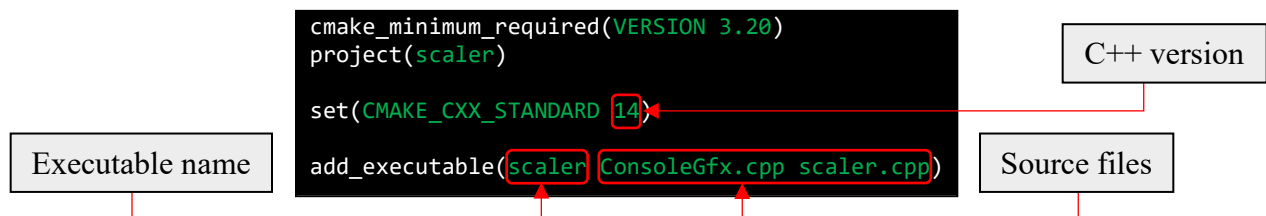
Changes the image data to the new pointer.

Building and Testing

For this project, students will begin to learn more details about the practical building and testing of C++ programs, including accessing executables built by the compiler from outside of the IDE.

Building

CLion uses the standardized CMake build system, which is supported across multiple platforms and IDEs (including CLion and Visual Studio) and supports many compilers (GCC, Clang, and MS Visual C.) When you create a CLion project, it will create CMakeLists.txt. The CMakeLists.txt file for this lab might look like this:



You should make sure that the CMakeLists.txt file correctly lists your source files and the executable. Then, click on the “Build” button (🔧), which is on the toolbar to the left of the “Run” (▶) and “Debug” (🐛) buttons.

Testing

Due to limitations of CLion’s (and “Command Prompt”) terminal, we will build our program using CLion but test using *Windows Terminal*. You should install it via the Windows Store. You will also need to add MinGW’s binary directory (`C:\ProgramData\chocolatey\lib\mingw\tools\install\mingw64\bin`) to the system’s PATH variable ([see Microsoft’s instructions](#)).

To open Explorer, open the build folder (e.g., `cmake-build-debug`), right-click on `scaler.exe`, and select **Open In → Explorer**. Then, right click in Explorer and select **“Open in Windows Terminal”**. Once the terminal window is open, you can run your program by typing the executable’s name (e.g., `scaler`):

```
Windows PowerShell
PS C:\Users\Jeremiah Blanchard\COP3504C\Lab 06\cmake-build-debug> .\scaler.exe
Welcome to the Image Scaler!

Displaying Spectrum Image:
[Color Spectrum Image]

Scaler Menu
-----
0. Exit
1. Load File
2. Load Test Image
3. Display Image
4. Enlarge Image
5. Shrink Image
6. Show Image Properties

Select a Menu Option: |
```

Submissions

NOTE: Your output must match the example output **exactly**. If it does not, ***you will not receive full credit for your submission!***

Files: `scaler.cpp`, `Image.cpp`, `Image.h`

Method: Submit on ZyLabs