CS 3005: Programming in C++

Color Table Class

Introduction

A color table is an array of colors. It is useful for translating a single number (index) in a range into a color (RGB values), reliably and repeatedly.

We will use a color table to translate Julia set and Mandelbrot set escape counts into colors to create colorful images.

Assignment

In this assignment you will create a class to store a single RGB [Color], and a class to store a vector of Color objects in a ColorTable. You will also modify your project to use these classes to provide users the ability to configure and use colors of their own choosing when creating PPM images from fractal escape values.

The ppm_menu program needs to add a few new commands.

The new commands required are:

- grid-apply-color-table) Use the grid values to set colors in the output image using the color table.
- set-color-table-size) Change the number of slots in the color table.
- set-color) Set the RGB values for one slot in the color table.
- set-random-color) Randomly set the RGB values for one slot in the color table.
- set-color-gradient) Smoothly set the RGB values for a range of slots in the color table.

Potential Session

```
# To run all of the commands from a script, throwing away the prompts
$ ./ppm_menu < ppm_menu_assignment_10_sample_session_color_table.txt >> /dev/null
$ ls -l *.ppm
```

```
-rw-r--r-- 1 cgl cgl  750015 Mar 14 09:02 sample-color-image-1.ppm
-rw-r--r-- 1 cgl cgl  3000017 Mar 14 09:02 sample-color-image-2.ppm
-rw-r--r-- 1 cgl cgl  3000017 Mar 14 09:02 sample-color-image-3.ppm
```

Programming Requirements

Below, the functions and methods may have a symbol CG? before them, where ? is a number. This indicates which Code Grinder step requires the function or method to be implemented.

Create ColorTable.h and ColorTable.cpp

These files will be used to declare and define both the Color and the ColorTable classes.

Color class

The Color class needs to store the following data:

- The integer representation of red, green and blue channels of a color.

The Color class needs to have the following methods:

- Color(); Sets all color channels to value 0.
- Color( const int& red, const int& green, const int& blue ); Sets the color channels to the values provided here. No range checking is applied.
- int getRed( ) const; Returns the value of the red channel.
- int getGreen( ) const; Returns the value of the green channel.
- int getBlue( ) const; Returns the value of the blue channel.
- int getChannel( const int& channel ) const; Returns the value of the channelth channel. 0 == red, 1 == green, 2 == blue. Returns -1 if the channel is out of range.
- void setRed( const int& value ); Changes the red channel to value. If value is less than 0, do not make any changes.
- void setGreen( const int& value ); Changes the green channel to value. If value is less than 0, do not
Add this method:

```cpp
void setPPM(PPM& ppm, const ColorTable& colors) const;
```

You will add a method to set a PPM object from the grid numbers, using a `ColorTable` instead of the built in table with 8 colors. Do not remove the previous method. Just add this one. Note, you may need to add an `#include`

Add this method:

```cpp
void setPPM(PPM& ppm, const ColorTable& colors) const;
```

Uses the currently stored grid numbers to configure an image in the PPM object. Sets the width and height of the image to match the width and height of the grid. Sets the maximum color value to the maximum color value of any color in the color table (`getMaxChannelValue()`). For each pixel in the PPM object, sets the color based on the grid number for the pixel. If the color table does not have at least 3 colors, make no changes to the PPM object. Use
the color table item at the last position for any pixels with a maximum number. Use the color table item at the next to last position for any pixels with a 0 grid number. For all other grid numbers, use (number) mod (the size of the color table - 2) as the index into the color table.

**Update** image_menu.h and image_menu.cpp

Add the following function declarations to the header file and implementations to the .cpp file.

- void applyGridColorTable( std::istream& is, std::ostream& os, NumberGrid& grid, ColorTable& table, PPM& dst ); Uses the new setPPM method to set the PPM using the grid and color table. Note this is not a replacement for applyGrid, this is in addition to that function.
- void setColorTableSize( std::istream& is, std::ostream& os, ColorTable& table ); Asks the user for the “Size? “, then applies it to the table.
- void setColor( std::istream& is, std::ostream& os, ColorTable& table ); Asks the user for “Position? “, “Red? “, “Green? “, and “Blue? “. The uses them to set a color at the specified position in the color table.
- void setRandomColor( std::istream& is, std::ostream& os, ColorTable& table ); Asks the user for “Position? “, then uses setRandomColor() to set a random color at that position. Use 255 for the maximum color value.

The following functions will require updates to their functionality and/or declarations.

- void showMenu( std::ostream& os ); Add to the menu to include the messages shown in the description above.
- void takeAction( std::istream& is, std::ostream& os, const std::string& choice, PPM& input_image1, PPM& input_image2, PPM& output_image, NumberGrid& grid, ColorTable& table ); Add clauses to recognize the 5 new commands and call the correct function for each one. Also, note the addition of ColorTable& table as the last parameter.
- int imageMenu( std::istream& is, std::ostream& os ); Add the declaration of a ColorTable object before the while loop. This object should have 16 colors in the table, and should be filled with a gradient from 0,255,0 to 255,0,255. It will then be passed to takeAction() inside the loop.

**Update** ppm_menu.cpp

No changes are required for ppm_menu.cpp.

**Update** Makefile

- This file must include the rules to build the program ppm_menu.
- A developer must be able to use the command make to compile all necessary files and link them to the executable program ppm_menu.
- The all target is expected in the Makefile.
- The clean target is expected in the Makefile.
- Automatic source and object file calculation are expected in the Makefile.
- Automatic dependency calculations are expected in the Makefile.

**Additional Documentation**

- C++ Reference
- Examples from class
- Sample Session Input File
- Julia set on Wikipedia
- Mandelbrot set on Wikipedia
- Color Gradient on Wikipedia (Only marginally useful.)
- Color Gradient Discussion
- Hints on choosing color schemes
- Paletton color selection site

**Sample PPM Images**

- Sample Output1
- Sample Output2
Show Off Your Work

To receive credit for this assignment, you must

- complete the unit tests available in CodeGrinder
- zip the source code (.cpp and .h files) and the Makefile and upload to the Canvas submission system
- use git to add, commit and push your solution to your repository for this class.

Additionally, the program must build, run and give correct output.

Extra Challenges (Not Required)

- Create additional methods in the `ColorTable` class that allow for easy insertion of interesting color patterns. Add the ability to use them from the `imageMenu()`. For example, can you implement a system to use a color and its complement to make a gradient?
- Create a method of `NumberGrid` to find the maximum number stored. Add the ability to set the number of colors in the color table to match this number from `imageMenu()`.
- Try other ways to modify the color system to make good color systems. For example, can you make an HSV based color system that would make setting the color more convenient for designers?