CS 3005: Programming in C++

Image Filters

This assignment requires extending the text-based application for working with images. You will now be able to filter images using some grayscale filters, and adding some primitive 2D shapes.

Assignment

In this assignment, you will update the program from the previous assignments to allow the user to create PPM images by filtering images with grayscale conversions, and by adding geometric shapes to images. All of the previous assignment's functionality will remain intact.

The linear colorimetric conversion formula is: brightness = 0.2126*red + 0.7152*green + 0.0722*blue.

Programming Requirements

The following files must be updated or created and stored in the `src` directory of your repository.

Make changes as described below.

Update `PPM.{h,cpp}`

The following methods must be declared and implemented for the `PPM` class.

- `void grayFromChannel( PPM& dst, const int& src_channel ) const;` Configures the meta-data of `dst` to be the same as the meta-data of `*this`. The meta-data includes the height, width, and maximum color value. For a given pixel in `*this`, copy the `src_channel` channel value into all three channels of the same pixel of `dst`. For example, if the pixel of at row 3 and column 7 of `*this` has a blue channel value of 18, and `src_channel` is 2, then the pixel at row 3 and column 7 of `dst` will have red, green, and blue channel values set to 18.
- `void grayFromRed( PPM& dst ) const;` Calls `grayFromChannel` to set `dst` from the red channel.
- `void grayFromGreen( PPM& dst ) const;` Calls `grayFromChannel` to set `dst` from the green channel.
- `void grayFromBlue( PPM& dst ) const;` Calls `grayFromChannel` to set `dst` from the blue channel.
- `double linearColorimetricPixelValue( const int& row, const int& column ) const;` Calculates the linear colorimetric value for the pixel at `row`, `column`, and returns it.
- `void grayFromLinearColorimetric( PPM& dst ) const;` Sets `dst` to have the same meta-data as `*this`. Sets every pixel in `dst` to have all channels (Red, Green and Blue) set to the linear colorimetric value calculated for the pixel at the same location in `*this`.

Update `image_menu.h` add `image_filters.cpp`

The follow functions must be declared and implemented.

- `void grayFromRed(ActionData& action_data);` Sets the output image to be the red to grayscale filtered copy of input image 1.
- `void grayFromGreen(ActionData& action_data);` Sets the output image to be the green to grayscale filtered copy of input image 1.
- `void grayFromBlue(ActionData& action_data);` Sets the output image to be the blue to grayscale filtered copy of input image 1.
- `void grayFromLinearColorimetric(ActionData& action_data);` Sets the output image to be the linear colorimetric grayscale filtered copy of input image 1.

Update `image_menu.h` add `image_drawing.cpp`

The follow functions must be declared and implemented.

- `void drawCircle(ActionData& action_data);` Asks the user for "Center Row? ", "Center Column? ", "Radius? ", "Red? ", "Green? ", and "Blue? ". Then fills in a circle shape with the color specified by the red, green and blue. All pixels that are no more than `radius` pixels from the center should be set. Distance is calculated as the square root of the sum of row difference squared and column difference squared. You could `#include <cmath>` and use `std::sqrt( )` to calculate the square root, or use the math trick shown in class. Use multiplication (`*`) to square values. Note that `std::sqrt()` will return a `double` value, so use the correct variable type to store the result. Make changes to the input image 1.
- `void drawBox(ActionData& action_data);` Asks the user for "Top Row? ", "Left Column? ", "Bottom Row? ",

"Right Column?", "Red? ", "Green? ", and "Blue? ". Then fills in a rectangle shape with the color specified by the red, green and blue. All pixels that have a row between the top and bottom row (inclusive) and between the left and right column (inclusive) should be set. Make changes to the input image 1.

Update `controllers.cpp`

The following functions will require updates to their implementations.

- `void configureMenu( MenuData& menu_data )` add the new actions with the names and descriptions listed below.

Table of New Commands

Command Name	Function Name	Description
red-gray	grayFromRed	Set output image by grayscale from red on input image 1.
green-gray	grayFromGreen	Set output image by grayscale from green on input image 1.
blue-gray	grayFromBlue	Set output image by grayscale from blue on input image 1.
linear-gray	grayFromLinearColorimetric	Set output image by linear colorimetric grayscale on input image 1.
circle	drawCircle	Draw a circle shape in input image 1.
box	drawBox	Draw a box shape in input image 1.

Update `Makefile`

This file must now also include a rule for `clean`. The following commands should work correctly.

- `make hello` - builds the hello program
- `make questions_3` - builds the questions_3 program
- `make ascii_image` - builds the ascii_image program
- `make image_file` - builds the image_file program
- `make ppm_menu` - builds the image_file program
- `make all` - builds all programs
- `make` - builds all programs (same as `make all`)
- `make clean` - removes all .o files, and all executable programs

Additional Documentation

- C++ Reference
- Examples from class
- Grayscale on Wikipedia
- Digital Image Processing on Wikipedia

Sample PPM Images

- Color Test Pattern
- Color Test Pattern Red
- Color Test Pattern Green
- Color Test Pattern Blue
- Color Test Pattern Linear
- Starry Night
- Starry Night Red
- Starry Night Green
- Starry Night Blue
- Starry Night Linear Colorimetric

Show Off Your Work

To receive credit for this assignment, you must

- use git to add, commit and push your solution to your repository for this class.
- successfully pass all unit tests and acceptance tests

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

Extra Challenges (Not Required)

- Create additional image processing options.