CS 3005: Programming in C++

PPM Menu

The previous assignments have built up several functions that allow the ability to read `PPM` images from files, modify the images, and write images to files.

This assignment will create an application with a menu based interface that allows users to select actions on `PPM` objects. It will combine the previous work with a few new functions to make a text based image editing program.

Assignment

In this assignment, you will add a new program to the project. This program will have the image editing capabilities mentioned above.

It is expected that all previous programs will still work unchanged.

Programming Requirements

Now that you have the `ActionData` and `MenuData` classes, most functions will need to be updated as noted below. For each function, update the parameter list to match the list shown below. Be sure to do this in the header file as well as the implementation file. After updating the parameter list, edit the rest of the function to use the `ActionData` object instead of the previous parameters. For example use `action_data.getIS()` where you used to use `is`. A similar substitution will be needed to replace `os`. See the Additional Documentation section below for an example.

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

Update `user_io.cpp` and `image_menu.h`

Update these functions:

- `std::string getString( ActionData& action_data, const std::string& prompt );` Make the substitutions described.
- `int getInteger( ActionData& action_data, const std::string& prompt );` Make the substitutions described.
- `double getDouble( ActionData& action_data, const std::string& prompt );` Make the substitutions described.
- `int askQuestions3(ActionData& action_data)` Make the substitutions described.
- Any similar Exam 1 functions

Add these functions:

- `std::string getChoice( ActionData& action_data );` Uses `getString` to ask the user for a string, using "Choice? " as the prompt. Returns the value returned by `getString()`.
- `void commentLine( ActionData& action_data );` Uses `.read()` to read a single character at a time from the input stream. If the input stream is `.good()` after the read, and if the character read is not the newline character, repeat. Otherwise return. In other words, read from the input stream until the input stream has nothing to read, or a newline character is read. Do not do anything with the characters read.
- `void quit(ActionData& action_data);` Set the `ActionData` object to be done.

Update `image_drawing.cpp` and `image_menu.h`

Update these functions:

- `void diagonalQuadPattern( ActionData& action_data )` Make the substitutions described. Since the `Image` parameter has been removed, use `action_data.getInputImage1()` where you would have used the `image` parameter. For example, `action_data.getInputImage1().setHeight(height)`. Set the maximum color value to 255. (This was not necessary previously, because this function worked on an `Image`, which do not have maximum color values.)
- `void stripedDiagonalPattern( ActionData& action_data )` Make the substitutions described. Since the `PPM` parameter has been removed, use `action_data.getInputImage1()` where you would have used the `ppm` parameter. For example, `action_data.getInputImage1().setHeight(height)`.
- Any similar Exam 1 functions

Add these functions:

- `void setSize( ActionData& action_data );` Use `getInteger` to ask the user for the “Height? ” and “Width? “, and set the height and width of the input image 1.
- `void setMaxColorValue( ActionData& action_data );` Use `getInteger` to ask the user for the “Max color value? ” and set the maximum color value of the input image 1.
- `void setChannel( ActionData& action_data );` Use `getInteger` to ask the user for “Row? “, “Column? “, “Channel? “, and “Value? “, then the set the channel value in input image 1.
- `void setPixel( ActionData& action_data );` Use `getInteger` to ask the user for “Row? “, “Column? “, “Red? “, “Green? “, and “Blue? “, then the set the values in input image 1.
- `void clearAll( ActionData& action_data );` Set all pixels in input image 1 to have the color (0,0,0).

Update `image_output.cpp` and `image_menu.h`

Update these functions:

- `void drawAsciiImage(ActionData& action_data)` Make the substitutions described. Since the `Image` parameter has been removed, use `action_data.getOutputImage()` where you would have used the `image` parameter. For example, `action_data.getOutputImage().getHeight()`.
- `writeUserImage( ActionData& action_data )` Make the substitutions described. Since the `PPM` parameter has been removed, use `action_data.getOutputImage()` where you would have used the `ppm` parameter. For example, `action_data.getOutputImage().writeStream(fout)`.
- Any similar Exam 1 functions

Add these functions:

- `void copyImage(ActionData& action_data);` Sets the output image to be equal to the input image 1. Literally, the body contains: `action_data.getOutputImage() = action_data.getInputImage1();`. That’s it.
- `void readUserImage1( ActionData& action_data );` Uses `getString` to ask the user for the name of an existing PPM file to be read, using “Input filename? ” as the prompt. Opens the file as an `std::ifstream`, then uses `readStream()` to read the file into the input image 1. If the file does not open correctly, report to the that the file could not be opened. For example, if the file was named “foo.ppm”, then the message should be “‘foo.ppm’ could not be opened.”

Update `controllers.cpp` and `image_menu.h`

In each of the `assignment*()` functions create an `ActionData` object passing `is` and `os` to its constructor. Then, update all of the function calls in to pass the `ActionData` object as appropriate. If the function calls a function that modifies input image 1, then calls another that reads output image, copy the input image 1 to the output image between those calls. See the examples from the Additional Documentation section.

Update these functions:

- `int assignment1(std::istream& is, std::ostream& os)`
- `int assignment2(std::istream& is, std::ostream& os)` Copy input image 1 to output image between drawing and output.
- `int assignment3(std::istream& is, std::ostream& os)` Copy input image 1 to output image between drawing and output.
- Any similar Exam 1 functions.

Add these functions, they will support the menu driven application for editing images. They include `imageMenu()` the main loop that controls the new application.

- `void showMenu( MenuData& menu_data, ActionData& action_data );` For each command that was added to `MenuData` via `addAction()`, displays one line of text to the output stream of the `ActionData`. The lines are formatted like this: “command-name) command description”. See the ShowMenu() example below.
- `void takeAction(const std::string& choice, MenuData& menu_data, ActionData& action_data);` Uses `choice` as a command name to get a `ActionFunctionType` from the `MenuData`. If the function returned is not `0`, then call the returned function, passing the `ActionData` as its parameter. Otherwise, if the choice was “menu”, call `showMenu`. Otherwise (if the function was `0` and `choice` was not “menu”), display a message with the format: “Unknown action ‘bad-choice’.”, where `bad-choice` should be the `choice`.
- `void configureMenu( MenuData& menu_data );` Calls `addAction` on the `MenuData` object to add the commands listed below in the Table of Commands, their functions, and their descriptions.
- `int imageMenu(std::istream& is, std::ostream& os);` Creates an `ActionData` object with `is` and `os` used for its input and output streams. Creates a `MenuData` object. Uses `configureMenu` to configure the commands in the `MenuData` object. Uses a loop that will continue as long as the `ActionData` object is not “done” and the `ActionData` object’s input stream is `.good()`. The body of the loop will use `getChoice` to get the user’s command choice, and `takeAction` to execute the user’s command choice. Returns `0`.

Table of Commands

Command Name	Function Name	Description
draw-ascii	`drawAsciiImage`	“Write output image to terminal as ASCII art.”
write	`writeUserImage`	“Write output image to file.”
copy	`copyImage`	“Copy input image 1 to output image.”
read1	`readUserImage1`	“Read file into input image 1.”
`#`	`commentLine`	“Comment to end of line.”
size	`setSize`	“Set the size of input image 1.”
max-color-value	`setMaxColorValue`	“Set the max color value of input image 1.”
channel	`setChannel`	“Set a channel value in input image 1.”
pixel	`setPixel`	“Set a pixel’s 3 values in input image 1.”
clear	`clearAll`	“Set all pixels to 0,0,0 in input image 1.”
quit	`quit`	“Quit.”

`ShowMenu()` Example

If the following actions have been added to the `MenuData` via `addAction()`:

```
menu_data.addAction("a", a_one_function, "A-One");
menu_data.addAction("b", balsamic_function, "Balsamic");
menu_data.addAction("c", chipotle_function, "Chipotle");
menu_data.addAction("d", dijon_function, "Dijon");
```

then the expected output would look like this:

```
a) A-One
b) Balsamic
c) Chipotle
d) Dijon
```

Create `ppm_menu.cpp`

This file must include the implementations of the following functions:

- `int main( );` This function should call `imageMenu`, passing in `std::cin` and `std::cout` as the input and output streams to use. This function should return what `imageMenu` returns.

Update `Makefile`

This file must now also include the rules to build the program `ppm_menu`. 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 ppm_menu program
- `make` - builds all programs

Additional Documentation

- C++ Reference
- Examples from class
- Windows PPM Viewer
- Linux PPM Viewer: `eog file.ppm`
- `ActionData` Examples

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 programs must build, run and give correct output.