Image Filter Program Using Object-Oriented Programming

Jongwon Yang, Kijae Kim, Sunmin Lee, and Yeonwoo Kook

School of Computer Science and Engineering, Chung-ang University

Object-Oriented Programming

Professor Bong-Soo Sohn

December 7, 2021

Table of Contents

Summary

In this project, we developed an image filter program using OpenCV. Our image filter program can manage your images and apply various filters to your images. You can also manage filters, and you can create and save custom filters. A custom filter is a filter that receives values such as brightness, contrast, and sharpness from the user and makes the desired settings.

This project is not only a useful program in real life, but also actively applied concepts such as message passing, inheritance, polymorphism, and dynamic binding by applying object-oriented programming.

It was also a good project to consider code reuse and extensibility through the use of libraries.

Compilation

For Windows

System Requirements

1. Visual Studio 2019 or higher
2. C++17 or higher
3. OpenCV 4.5.3 or higher

Instructions

1. Open source code folder for Windows.
2. By clicking “Project1.sln” file, open Visual Studio project.
3. Make sure your Visual Studio is ready to compile with C++17 or higher.
    a. Go to “Project → Properties → C/C++ → Language → C++ Language Standard”.
    b. Set it to “ISO C++17 Standard”.
4. Make sure your Visual Studio is ready to use OpenCV library.
    a. Download OpenCV for Windows from https://opencv.org/releases/ and install.
    b. In your Visual Studio project, right click the project name and open “Properties”.
    c. Set “Configuration” to “All Configurations”.
    d. Set “Platform” to “x64”.
    e. Go to “C/C++ → Additional Include Directories” and add the “build\include” directory of installed OpenCV location.
        i. e.g. “C:\opencv\build\include”

f. Go to “Linker → Additional Library Directories” and add the “x64\vc15\lib” directory of installed OpenCV location.
    i. e.g. “C:\opencv\build\x64\vc15\lib”
g. Go to “Linker → Input → Additioanl Dependencies” and add “.lib” files in the lib folder.
    i. e.g. “opencv_world454.lib;opencv_world454d.lib;”
h. Go to “Configuration Properties → Debugging → Environment” and set it to “build\x64\vc15\bin;%PATH%” directory of installed OpenCV location.
    i. e.g. “PATH=C:\opencv\build\x64\vc15\bin;%PATH%”.
i. Apply and close.
j. Set “Solution Configuration” to “Release”.
k. Set “Solution Platform” to “x64”.
l. At this point, the C++ version settings are sometimes reset. In that case, please set the C++ version to 17 again.

5. Press “Ctrl + Shift + B” to build.

For macOS

System Requirements

1. Xcode 9.3 or higher (12 is recommended)
2. C++17 or higher
3. OpenCV 4.5.3 or higher

Instructions

1. Open source code folder for macOS.
2. By clicking “ImageFilter.xcodeproj” file, open Xcode project.

3. Make sure your Xcode is ready to compile with C++17 or higher.
    a. Go to “Project → Build Settings → Apple Clang – Language – C++ → C++ Language Dialect”
    b. Set it to “GNU++17”.
4. Make sure your Xcode is ready to use OpenCV library.
    a. Install “Homebrew” and “Git”.
        i. “Homebrew” – https://brew.sh/
        ii. “$ brew install git”
    b. Install “OpenCV” through “Homebrew”. This may take some time.
        i. “$ brew install opencv”
    c. Install “pkg-config” through “Homebrew”.
        i. “$ brew install pkg-config”
    d. Extract link flag through “pkg-config”.
        i. “$ pkg-config --cflags --libs opencv4”
        ii. It will print the link flag as a result. You need to copy this.
    e. In your Xcode project, go to “Project → Build Settings” tab.
    f. Search for “Header Search Path” and set it to “opencv4” directory inside “include” folder of installed OpenCV location
        i. e.g. “/opt/homebrew/Cellar/opencv/4.5.3_3/include/opencv4” or “/usr/local/Cellar/opencv/4.5.3_3/include/opencv4”
    g. Search for “Library Search Path” and set it to “lib” directory of installed OpenCV location.

i. e.g. “/opt/homebrew/Cellar/opencv/4.5.3_3/lib” or “/usr/local/Cellar/opencv/4.5.3_3/lib”

h. Search for “Other Linker Flags” and set it to your linker flag (You’ve copied this above).

i. Go to “Targets → Signing & Capabilities → Hardened Runtime” and check “Disable Library Validation”

5. Press “command + B” to build.

Execution

If you compiled yourself, you need to create an "images" folder and a "customs.txt" file in the location of the executable to run it.

For Windows

If your compilation was successful, you can execute the program directly in Visual Studio by pressing “Ctrl + F5”.

Or you can double-click "Project1.exe" in the "executable/Windows" directory to run it. However, the “Project1.exe” file was built in Windows 10 64-bit, x64-based proccessor. So it may not be able to run if the environment is different.

For macOS

If your compilation was successful, you can execute the program directly in Xcode by pressing “command + R”. However, in the IDE console environment, camera permission cannot be obtained, so the camera function of the program cannot be used. So, after building in Xcode, go to the location of the executable directly in the terminal and run it.

Or you can directly run “ImageFilter” in the “executable/Mac” directory using terminal. In this case, if you grant the camera permission only once, the camera function works normally.

But, the “ImageFilter” file was built in macOS 11 64-bit, ARM-based M1 proccessor. So it may not be able to run if the environment is different.

Functionality

Manage Filter

In “Manage filter” menu, you can list basic (built-in) filters. You can also create, read, update, delete custom (user-defined) filters. The following is a description of each menu.

1. List basic filters: It prints list of basic filters.
2. List custom filters: It prints list of custom filters.
3. Create custom filter: You can create user-defined filter interactively.
4. Show custom filter: You can choose an existing custom filter and see it’s detail.
5. Edit custom filter: You can choose an existing custom filter and edit it.
6. Delete custom filter: You can delete an existing custom filter.

Manage Image

In “Manage image” menu, you can manage images in your folder. And you can apply filters to the images. You can also take a picture using the webcam, and save it into image folder.

1. List images: It prints list of images in “images” folder.
2. Show image: You can choose an image and see it.
3. Apply filter: You can choose an image and apply a filter to it.
4. Take photo: You can take a picture using webcam.
5. Delete image: You can delete an image.

Implementation

Applying a Filter to an Image

"Filter" is an abstract class containg a pure virtual function – *apply*. The *apply* method takes an *Image* pointer as a parameter. All filters must extend the *Filter* class. That is, you must implement the *apply* method. In the *apply* method, you can apply the desired effect to the image by using various functions of OpenCV. The basic filter created in this way is built-in as 6 classes – Grayscale, Warm, Cool, Sepia, Pencil, InvertFilter.

```
// virtual method in Filter class
virtual void apply(Image *image) = 0;
```

By doing this, in the *applyFilter* method of the *Image* class that receives the *Filter* pointer as a parameter, the unified action of calling the *apply* method can be implemented regardless of the class of the actually entered filter.

```
// applyFilter method in Image class
void applyFilter(Filter *filter);
```

Creating Custom Filter

CustomFilter is also a class that inherits from *Filter*. So you need to implement the *apply* method. However, since the user cannot interactively modify the code inside the *apply* method, we have applied numerical values to several predefined effects such as, brightness, contrast, blur, sharpness, etc. By saving these values as a file line by line, you can manage custom filters created by users.

```
// Example of line saved in customs.txt file
Let there be light;100;0;0;0;0
```

Listing Images in Folder

We used the standard library *filesystem* provided from C++17. The *filesystem* library provides an *iterator* that can search the list of files in a specific folder. In the constructor of the

ImageManager class, the list of files in the "./images" folder was called using the *iterator*. The list of loaded files was managed as a member variable using *vector*.

```
// Example of using filesystem iterator
for (const auto& entry : fs::directory_iterator(path)) { … }
```

Taking photo

It seems quite difficult to use the laptop's camera using C++. However, since OpenCV provides a class called *VideoCaputre*, productivity can be greatly improved by using a predefined library.

UML Diagram

Figure 1

Class Diagram for ImageFilter Program

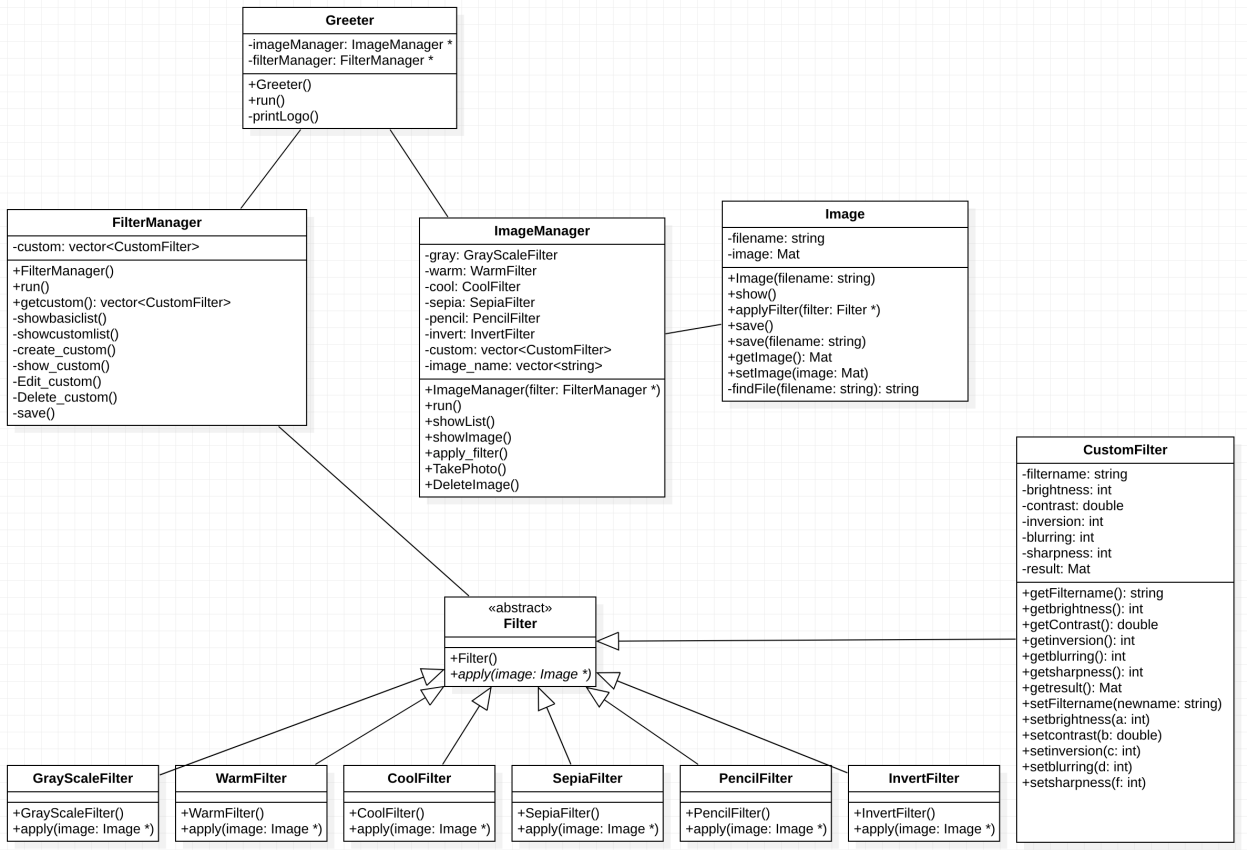


Figure 2

Use-Case Diagram for ImageFilter Program

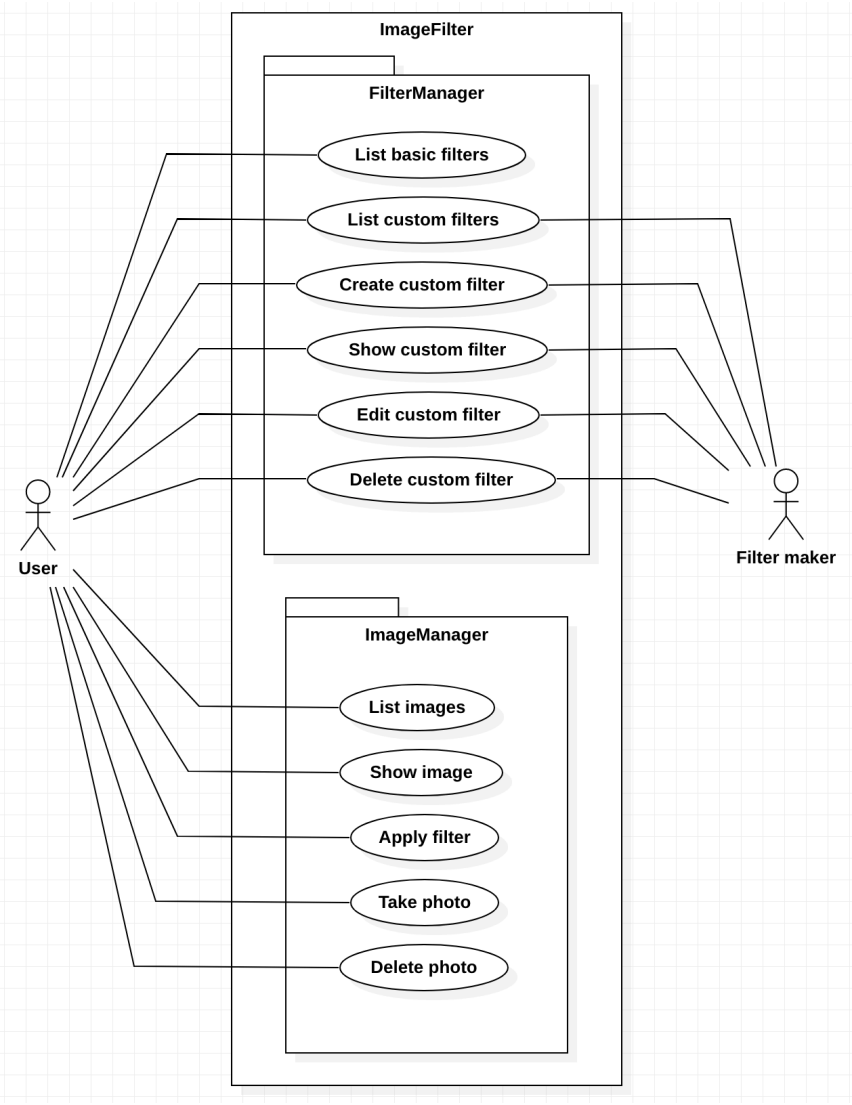
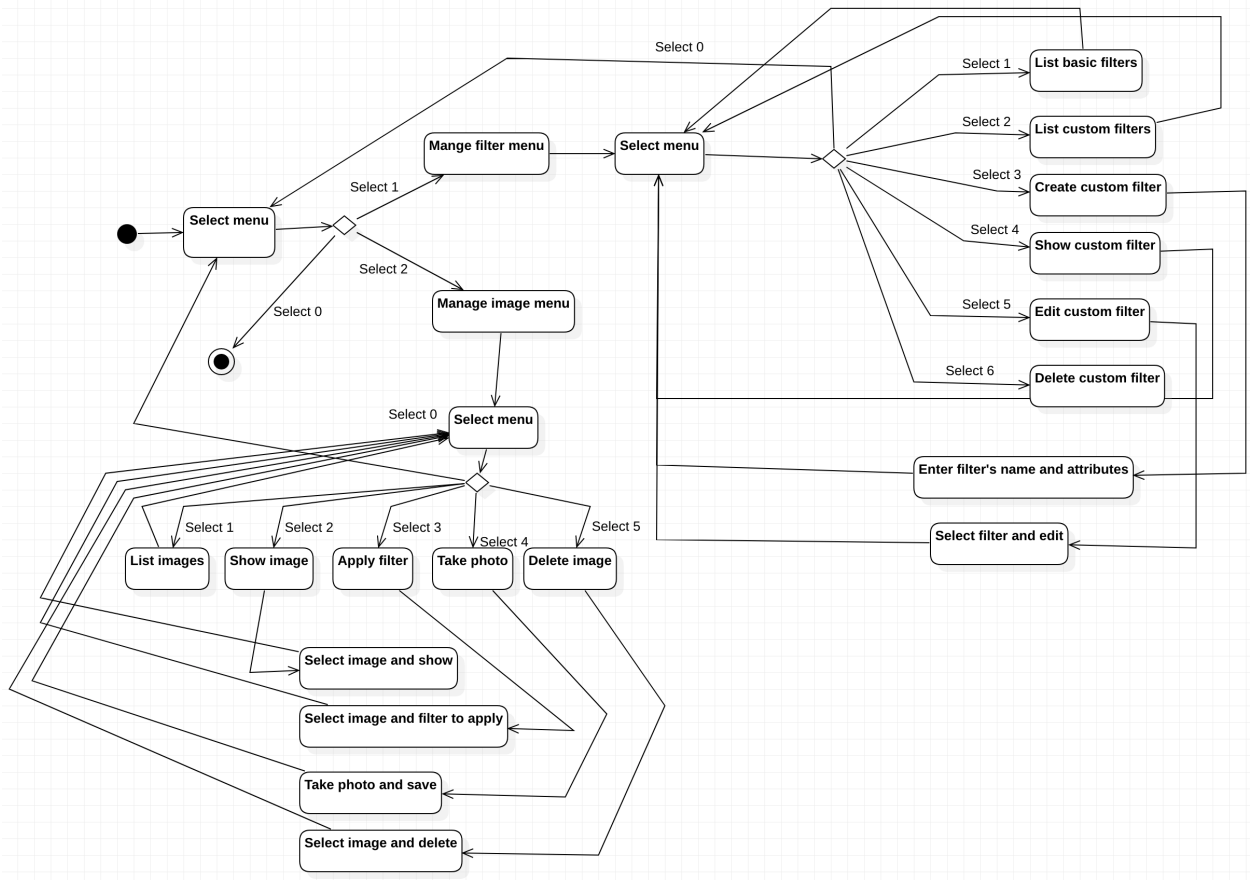


Figure 3

Activity Diagram for ImageFilter Program



Execution Result

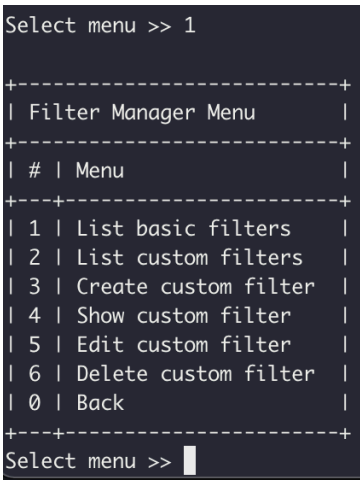
Figure 4

Screen Capture of Main Menu

Note. This figure is a capture of the main menu that appears first when the program is run.

Figure 5

Screen Capture of Filter Manager Menu



Note. This figure is a screen capture of the menu that appears when you select 1 in the main menu.

Figure 6

Screen Capture of Basic Filter List

Note. This figure is a screen capture of the menu that appears when 1 is selected in the Filter Manager menu.

Figure 7

Screen Capture of Custom Filter List

Note. This figure is a screen capture of the menu that appears when 2 is selected in the Filter Manager menu.

Figure 8

Screen Capture of Creating Custom Filter

```
Select menu >> 3
If you don't want to apply the effect, enter 0 in the corresponding element.
Insert filter name: New Filter
insert brightness (range -100 ~ 100) : 50
insert contrast (range 0 ~ 2, real number) : 1
insert inversion (range 0 ~ 3, 1: Left and right inversion, 2: Top and bottom inversio
n, 3: All of inversion) : 0
insert blurring (range 0 ~ 3) : 0
insert sharpness (range 0 ~ 2, 0: No apply, 1: Soft, 2: Hard) : 0
Complete!!
```

Note. This figure is a screen capture of selecting 3 from the Filter Manager menu and creating a new custom filter.

Figure 9

Screen Capture of Showing Custom Filter

```
Select menu >> 4
Filter name: New Filter
Brightness: 50
Contrast: 1
Saturation: 0
Exposure: 0
Shadow: 0
```

Note. This figure is a screen capture of selecting 4 in the Filter Manager menu and inquiring the newly created custom filter.

Figure 10

Screen Capture of Modifying Custom Filter

Note. This figure is a screen capture of selecting 5 from the Filter Manager menu and editing the contents of custom filter.

Figure 11

Screen Capture of Image Manager Menu

Note. This figure is a screen capture of the menu that appears when you select 2 in the main menu.

Figure 12

Screen Capture of Image File List

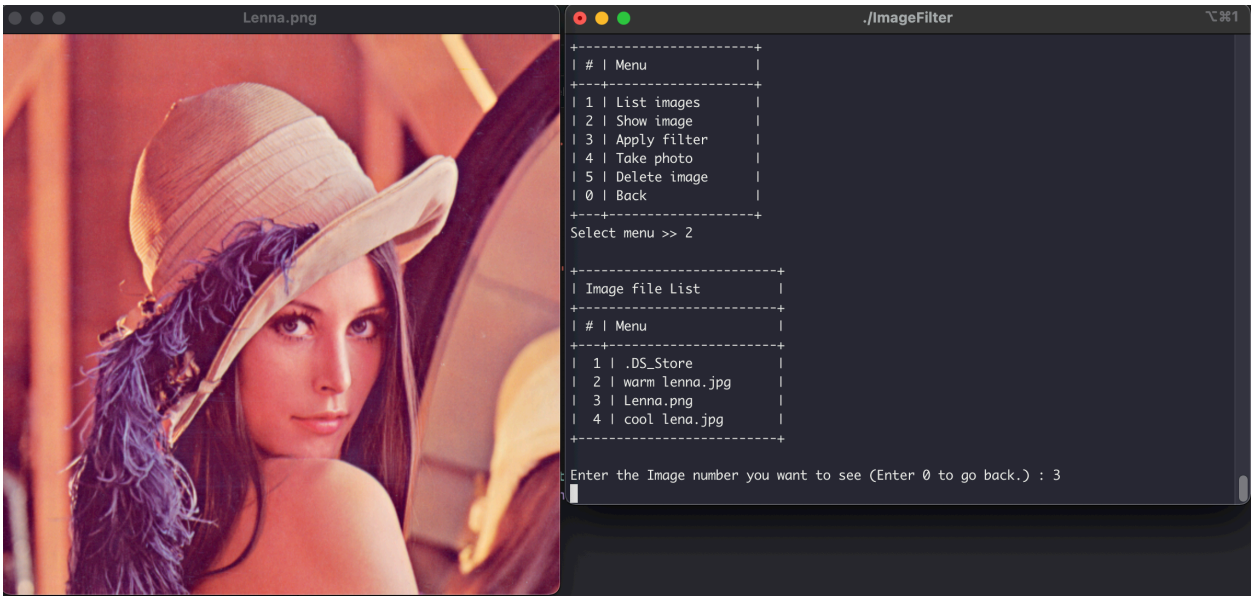
```
Select menu >> 1

+--------------------------+
| Image file List          |
+--------------------------+
| # | Menu                 |
+---+----------------------+
|  1 | .DS_Store           |
|  2 | warm lenna.jpg      |
|  3 | Lenna.png           |
|  4 | cool lena.jpg       |
+--------------------------+
```

Note. This figure is a capture of the screen output by selecting 1 in the Image Manager menu.

Figure 13

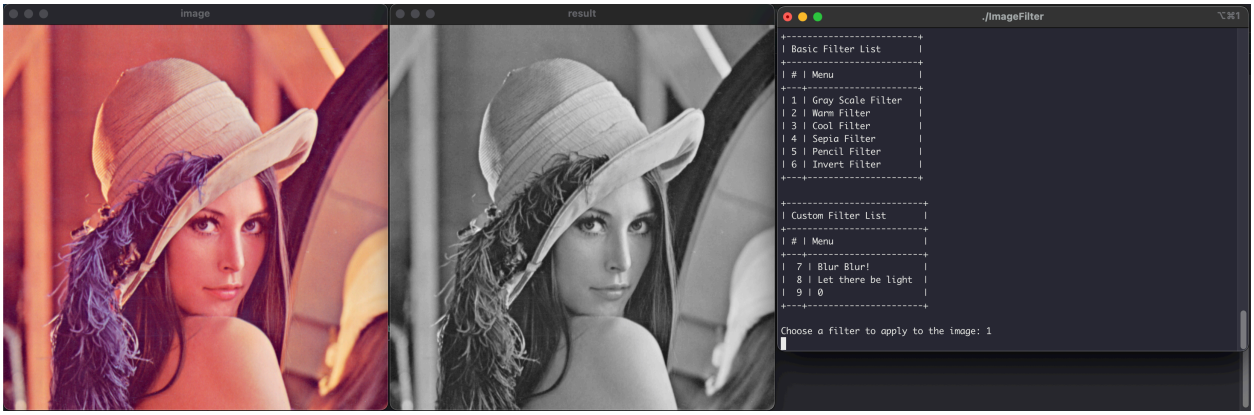
Screen Capture of Showing Image



Note. This figure is a screen capture showing the selected image by selecting 2 in the Image Manager menu.

Figure 14

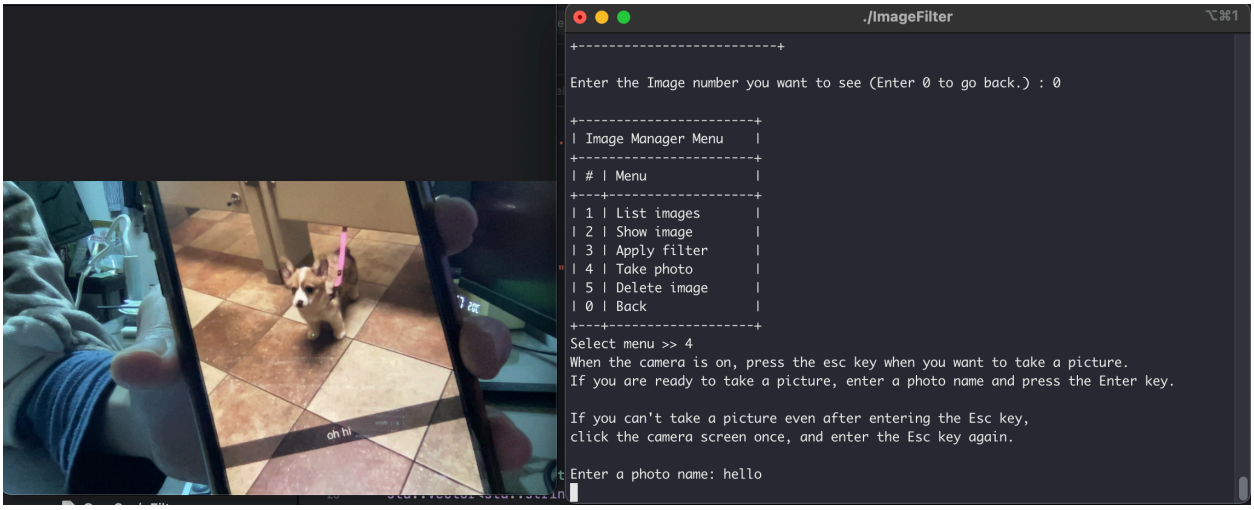
Screen Capture of Applying Filter



Note. This figure is a screen capture of selecting 3 from the Image Manager menu and applying the Grayscale filter.

Figure 15

Screen Capture of Taking a Picture



Note. This figure is a screen capture of selecting 4 in the Image Manager menu and taking a picture using the webcam.

Figure 16

Screen Capture of Deleting Image

Note. This figure is a screen capture of selecting 5 in the Image Manager menu and deleting the image.

Object-Oriented Concepts

The most basic units of our ImageFilter program are the *Image* class and the *Filter* class. In the *Image* class, by hiding the information of the actual image it contains, it was possible to clearly define the operations that can be performed on the image through *data encapsulation*. And the *Filter* class is an abstract class that contains pure virtual functions. The built-in basic filter and user-defined custom filter inherit this abstract *Filter* class. By doing so, we were able to unify the behavior of applying a filter to the image as it did for the *Filter* class. In other words, it is implemented to increase *code reusability* through *polymorphism*. This is because *dynamic binding* occurred when the virtual method apply was called. In addition, it is an implementation considering *extensibility* to allow users to create their own custom filter.

By separating the *FilterManager* class and the *ImageManager* class, it is advantageously used for collaboration through *control passing* . The person implementing the parent menu was able to collaborate without knowing the child implementation by using only one method to pass control to the child menu.

OpenCV was used in this project. Manipulating binary images directly in C++ or using hardware such as a webcam can be difficult. However, using a well-crafted library made it easy to implement this by taking advantage of *code reusability*. In addition, *standard libraries* such as vector and iterator were also very helpful for *productivity*.

Conclusion

Through this project, we felt the advantages of *object-oriented programming* while implementing it. We experienced *dynamic binding* based on *polymorphism* through *message passing*, and improved *reusability* and *extensibility* of code through *inheritance* and *polymorphism*.

And we also experienced *class-based operations* that can be useful in *collaboration*. It was very convenient that we did not have to worry about sub-menu implementation when implementing UI through *control passing*.

Finally, we experienced increased *code reusability* and *productivity* by using *libraries* such as *OpenCV* and *STL*. As developers, we felt that if we take advantage of object-oriented programming, we can use these libraries, or even create and provide them ourselves.