National University of Sciences And technology

December 2025

Technical Report

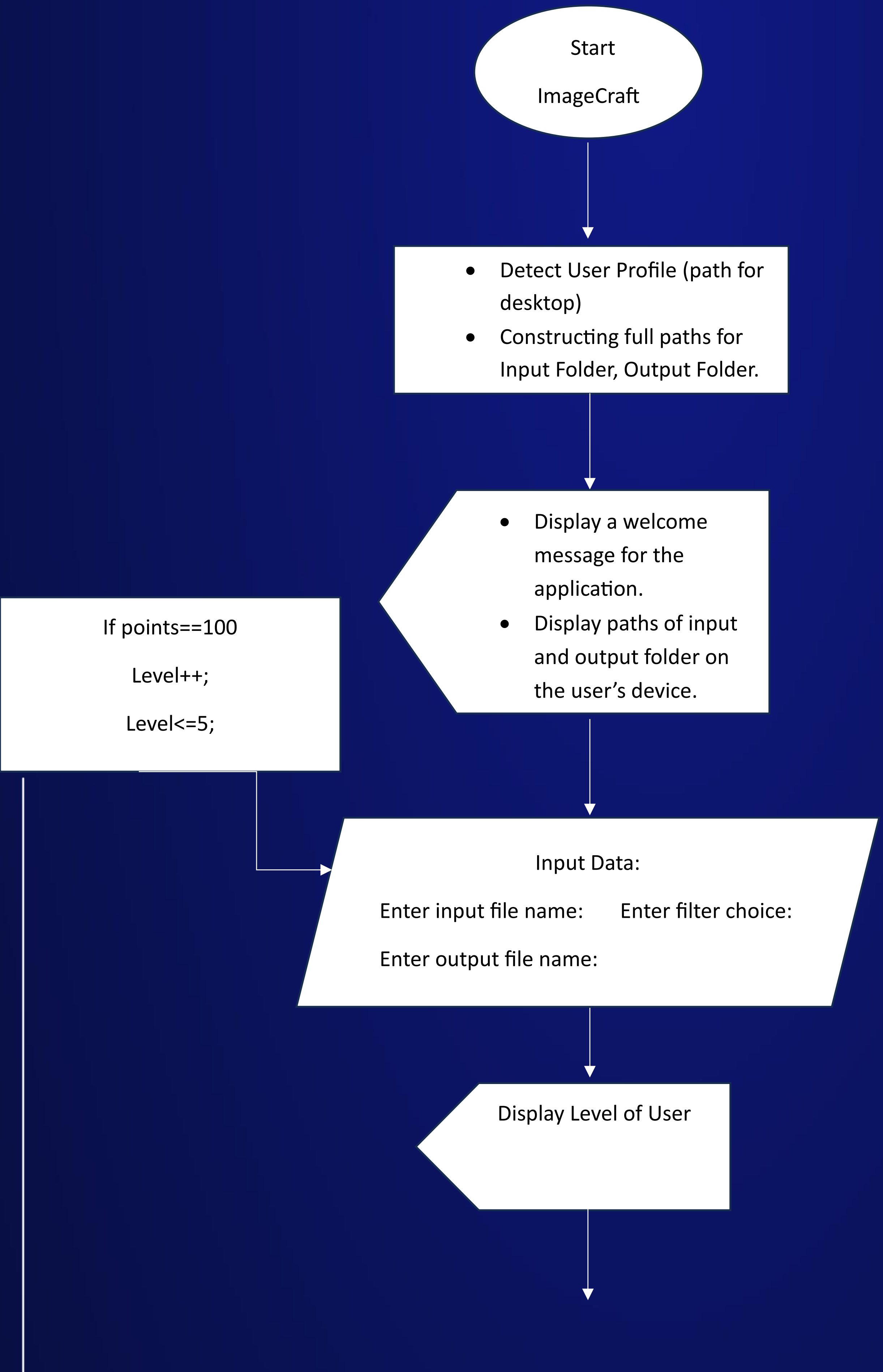
Image-Processing Game

Prepared By:

Fatima Faizan

Minahil Tahir

DESIGN LOGIC FOR IMAGECRAFT



Start

ImageCraft

- Detect User Profile (path for desktop)
- Constructing full paths for Input Folder, Output Folder.

- Display a welcome message for the application.
- Display paths of input and output folder on the user's device.

If points==100

Level++;

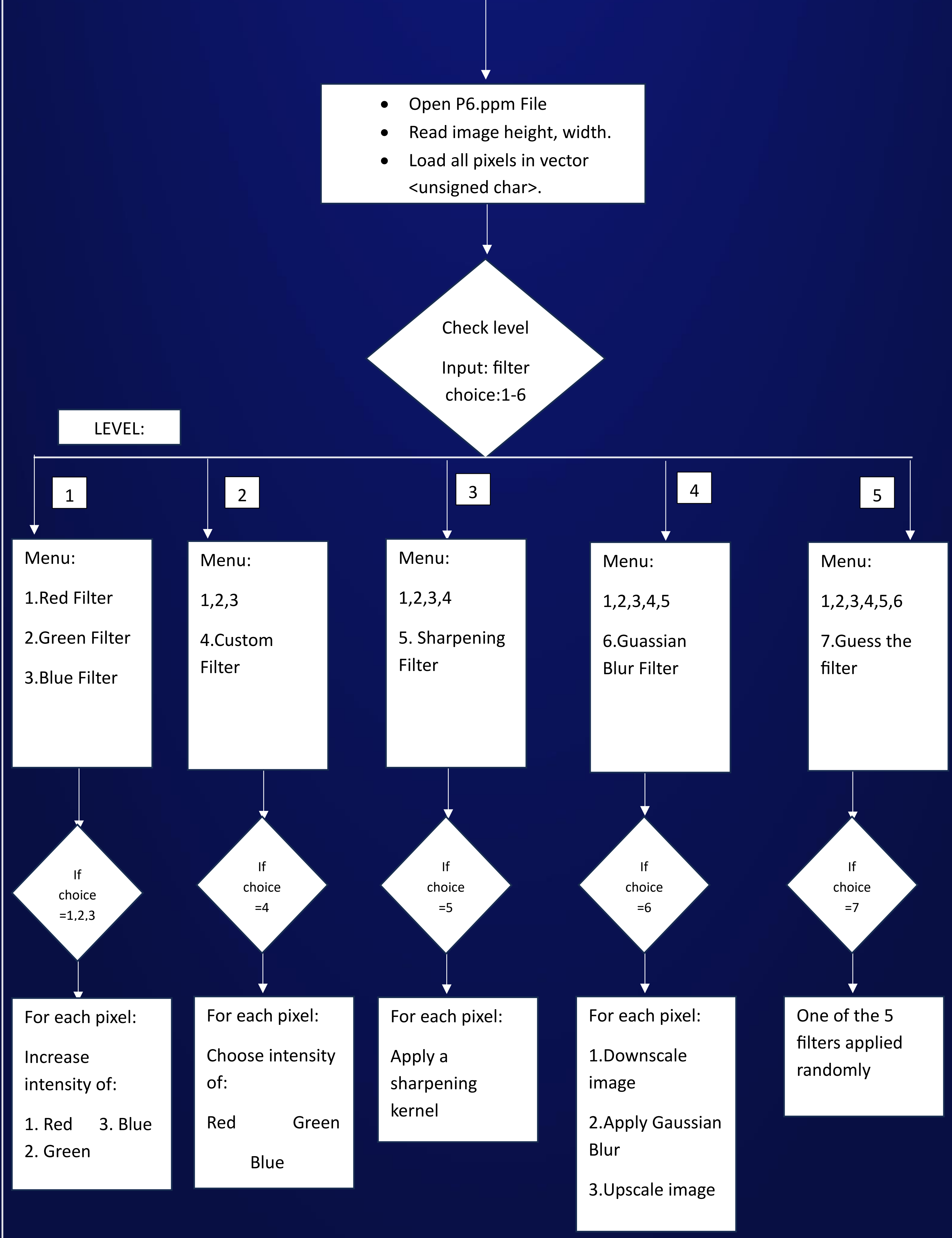
Level<=5;

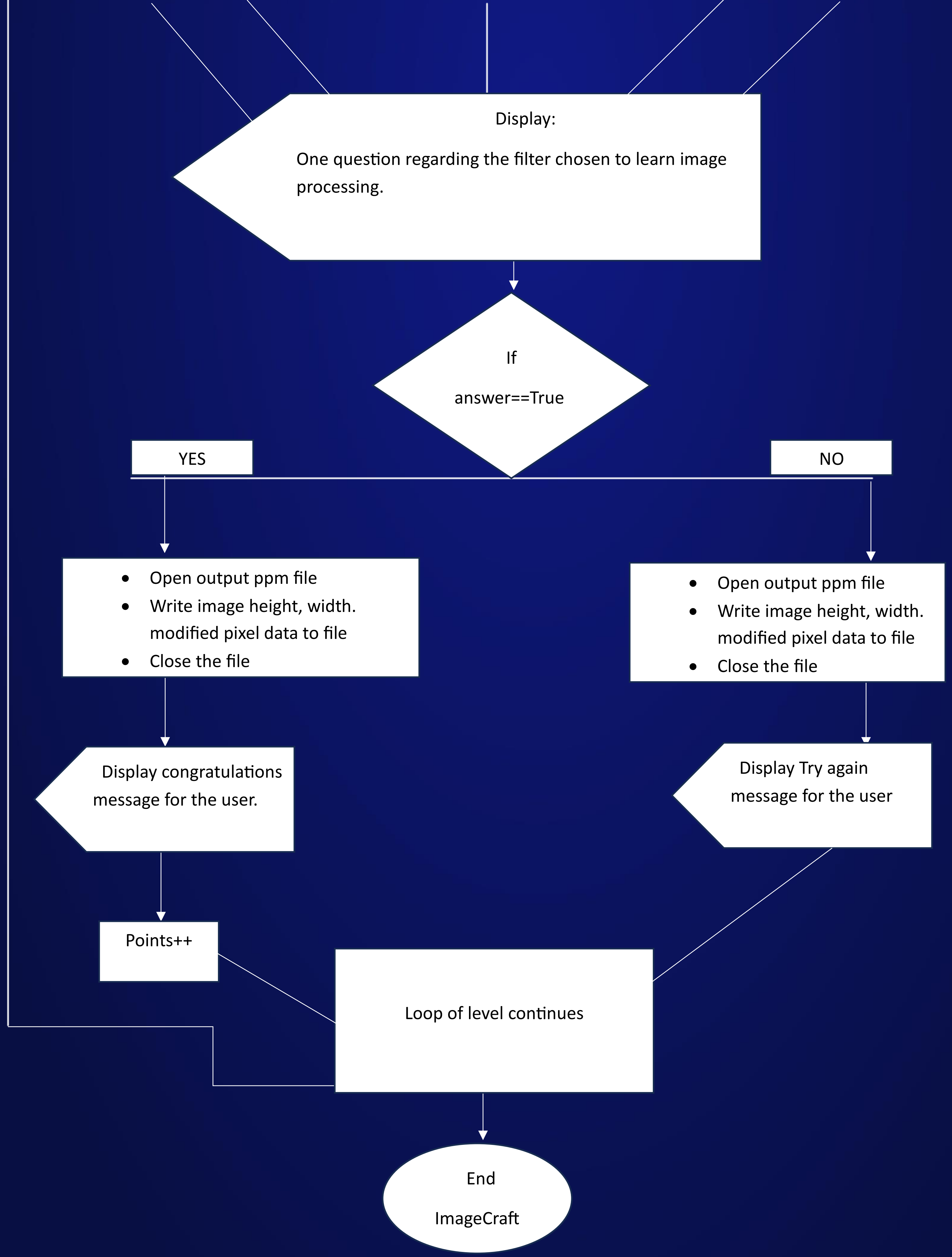
Input Data:

Enter input file name: Enter filter choice:

Enter output file name:

Display Level of User


Open P6.ppm File
Read image height, width.
Load all pixels in vector <unsigned char>.
Check level
Input: filter choice:1-6
LEVEL:
1
2
3
4
5
Menu:
1.Red Filter
2.Green Filter
3.Blue Filter
Menu:
1,2,3
4.Custom Filter
Menu:
1,2,3,4
5. Sharpening Filter
Menu:
1,2,3,4,5
6.Guassian Blur Filter
Menu:
1,2,3,4,5,6
7.Guess the filter
If choice =1,2,3
If choice =4
If choice =5
If choice =6
If choice =7
For each pixel:
Increase intensity of:
1. Red 3. Blue
2. Green
For each pixel:
Choose intensity of:
Red Green
Blue
For each pixel:
Apply a sharpening kernel
For each pixel:
1.Downscale image
2.Apply Gaussian Blur
3.Upscale image
One of the 5 filters applied randomly


Display:
One question regarding the filter chosen to learn image processing.
If
answer==True
YES
NO
• Open output ppm file
• Write image height, width. modified pixel data to file
• Close the file
• Open output ppm file
• Write image height, width. modified pixel data to file
• Close the file
Display congratulations message for the user.
Display Try again message for the user
Points++
Loop of level continues
End
ImageCraft

Algorithmic\Low-Level Design:

This program is an image processing application that applies various filters to the desired input image. It manipulates the image pixel data to achieve filtered effects like blurring, sharpening and color enhancing. We have included modularity by making functions of the filter functions and we've also gamified this experience by including levels for different filters and allowing you to level up. As you use more filters, you earn more points and unlock higher filters/levels.

Our program processes RGB images.

Each pixel contains 3 color components:

1. Red
2. Green
3. Blue

The concentration of each of these values can range from 0 to 255.

Images are stored in memory as vectors.

The total size of image is w*h*3.

(width into height into 3 for the number of color values per pixel)

Note:

Apart from that, we also offer a **custom filter** where you can adjust the values for red, green and blue in the range of 0-255. For example, one red pixel in an image is of value 200, we add 60 to it. That means 260>255.Hence, our program will consider it 255 since no pixel's value can be smaller than 0 or larger than 255.

Filtering:

Kernel:

It is the filter applied to photos and the filter application process works on convolution. Kernels are the 2D matrices which are aligned with pixels, multiplying them according to the kernel and adding their sums.

Gaussian Filter:

It uses a Gaussian 5x5 kernel (filter) that smoothens the image, reduces noise and gives a natural blur. It gives more weight to nearer pixels whereas lower weight to distant ones to reduce noise, all while preserving the natural image structure. It works by performing a weighted average of a pixel with its neighbors. The central pixels are given more weight as compared to the neighbors.

Sharpen Filter:

It enhances the sharp points and details in the image, which is contrary to what we did in the gaussian blur. This filter emphasizes differences between a pixel and its neighbors. Sharpening filter is used to improve picture clarity and clearness.

How our program flows:

1. Input image data is read into a pixel vector
2. A temporary out vector is created and processed pixels will be stored here.
3. The selected filter is applied using convolution process.
4. The processed image is stored in the out vector.
5. The final image is written back into an output file.

This process is very efficient, as no pixels are lost during the process.

Up and down scaling:

It is hard to filter a usual size photo by convoluting it with such filters and requires much computational power. An easier option is first downscaling it, applying the filter to the shrunk image and enlarging it back to the original size. The final photo has a strong filter effect with no casualties.

Downscaling: The width and height of the photo are reduced according to a fixed scale.

Upscaling: The image is brought back to the original size by upsizing the width and height according to the same scale.

Libraries:

2 new libraries are used in this program:

- ***fstream:*** used for file handling. It includes if steam and of stream, used for in and outputting files. (get(), read() from instream)

- ***vector:*** used for dynamic and contiguous memory allocation, basically dynamic arrays.

Tentative User-Interface:

1. **Console menu**: User will navigate through our program by either writing digits, choosing options through characters etc.
2. **Input prompts**: where the user enters a name for their input and output photo, answer for the question asked regarding image-processing and when they are choosing filters.
3. **Questions & feedback**: after applying a filter, image processing question is asked, user answers, and the program responds accordingly.
4. **Confirmation messages**: “Filter applied!”, “Image saved as output.ppm.”.
5. **Congratulations message**: If you answer correctly, you are awarded with points that will increase your level.
6. **Level System**: Depending on your points, you will level up and each level will unlock new filters for you to try.

Summary:

Overall, this program demonstrates how image processing works on the lower level. It provides insight to fundamental concepts in image processing by working on raw pixel data and kernels, using RGB values and using convolution to apply filters like gaussian blur, sharpening, etc.
It makes this experience enjoyable through the leveling of the filters.

Use-case diagrams:

The design flow-chart above has provided the use-case diagrams. To summarize following are the steps that will take place when the user will play ImageCraft:

1. Load Image: Player provides image file for processing along with naming the output file.

2. Apply Gaussian Blur: Smooth image with 5x5 kernel.

3. Apply Sharpen Filter: Enhance edges and details.

4. Apply Custom RGB: Adjust individual color channels.

5. Answer Quiz: Test image processing knowledge.

6. Earn Points: Score for correct answers.

7. Level Up: Unlock advanced filters.

8. Save Image: Export processed result.

Functional Requirements:

1. **Format:** program is able to read only .ppm formats.
2. **Gaussian Blur:** code will implement Gaussian blur using 5×5 kernel.
3. **RGB filters:** Program shall clamp RGB values to 0-255 range.
4. **Scoring:** Program shall track user points.
5. **Levelling:** Program shall unlock filters at specific point thresholds.
6. **Data saving:** Program shall save processed images as .ppm files.
7. **Error Handling:** Program shall handle invalid input or invalid choice.
8. **Questions:** Our code will ask quiz questions at the end of filter implementation.
9. **Data Persistence:** Program will save and load data of user (name, points, current level).
10. **Customization:** Code allows user to customize RGB values under the custom filter.
11. **Randomization:** Under the "guess the filter" option, user is given a random filter to guess. Apart from that, user will also be asked quiz questions regarding the filter randomly from the quiz question bank.

Non-Functional Requirements:

- **Preservation:** Shall preserve image integrity (no corruption during/after processing).
- **Pixel-loss:** Program guarantees **zero pixel-loss** unlike modern editors.
- **Speed:** As P6.ppm files are used, hence the process of filter application is extremely fast combined with downscaling and upscaling technique. If P3.ppm was used, the filter application would have taken more than 3-4 minutes.
- **Comments And Modularity:** Code is extremely simple to understand as it is fully commented and modular.
- **Learning Resource:** Code itself acts a source of teaching image-processing basics to our users as it is extremely easy to understand due to its modularity.

Tools And Techniques Used:

- **Input Folder:** ./input/ - Dedicated directory for source images
- **Output Folder:** ./output/ - Dedicated directory for processed images
- **Primary library: ImageMagick** (for converting formats of photos).
- **Convert PPM->JPG/JPG->PPM .bat files:** Drag and drop the image in this .bat file to convert the format.
- **Internal Format:** P6 PPM(binary) for fast pixel access.
- **Pixel Storage:** std::vector<Pixel> where Pixel = {r, g, b}
- **Kernel Representation:** 2D arrays for convolution matrices.

Future Enhancements:

Following enhancements can be made in our project:

- The menu giving an option to view stats at any point.
- An option to reset level in our game.
- A bigger question bank to assure questions do not repeat.
- More filtering options.
- Increase in levels.
- A more diverse program that supports other picture formats along with jpg.
- A timer can also be added to make the game challenging.

Git Repository Link:

https://github.com/minahilbese25seecs/ImageCraft-Minahil-Fatima.git