**“Clueless” Final Project Documentation**

**Team Sigma Delta Nu**

Suhani Mitra, Margherita Piana, Marissa Ruiz, Tara Gill

Department of Engineering, Boston University

EC 327: Introduction to Software Engineering

Dr. Douglas Densmore

December 12, 2022

Github Link: <https://github.com/suhanimitra/EC327Project.git>

YouTube Link:

**Overview:**

This program is an application of image processing, which is a growing area of research which is of interest to us. Our program is meant to prompt the user to input six P3 ppm files that correspond to an outfit, each file will be a component of the outfit (for example, one file is the hat, one the shirt, etc…). The program will then process each of them, read the RGB values of each pixel and find an average of each. There is a range for each RGB value that corresponds to a color, so for each image the dominant color will be found. The color compatibility of each component is then checked for all inputted files. The program will finally inform the user whether the outfit created is compatible or not, similar to how the “Clueless” program functioned.

**Marketability:**

This program is marketed toward users who are a fan of the movie “Clueless” and want to live Cher’s reality with her computerized closet. More seriously, our code is for users who are interested in potentially taking risks with their fashion, but want some quantitative validation on those risks. For example, users can explore different color, pattern and texture combinations in their outfits without actually having to step in public. This is because our app aims to read in the pictures of their clothes and tell the user if their virtual outfit is a “slay” or a “nay.” The app intends to help these users by looking at a user-inputted picture of a clothing article and determines if the outfit is compatible. The user will take pictures of individual articles of clothing and the program will detect whether the colors are aesthetically complementary, letting the user know if so/not.

**Component Description Part I (Front End):**

When developing our user interface, we focused on making it easy to use. We ensured that this was possible by creating clear, straightforward, distinct prompts for the user at every stage. We clearly labeled where the user should type in each file. Then the user is prompted to click an “enter” button, which is distinctly colored from its background, after they are done typing in their file names.

We created our user interface as a windows desktop app using Visual Studio 2022 and a public github repository called Dear ImGui, which references GUI or graphical user interface. This github repository is linked in references. The Dear ImGui repository has many examples for different API’s (Application Program Interface), but we decided to build our code off of the example using DirectX9 and Win32 because it is older and a little more simpler. The API used should not affect the ImGui code, it just affects the behind the scenes processing. We intentionally set the theme of the app to be clear and bright, emulating the personality of the movie this app is based on, Clueless. Using the ImGui libraries, we created two widgets (windows). The first window asks the user to input file names for each article of clothing (headwear, top, button, jacket, and shoes). There is also an option to opt out of an article of coding inputting “NA.” The second window displays a lava lamp graphic, which was created using a public code written for Dear ImGui by user heretique on github. This code is also linked in references. The lava lamp is an ode to the 90s, taking the user back to the days when Clueless came out. We changed the font to look a little bit older and pixel-esqe to resemble the computer app Cher used in the movie clueless. An image of this interface from the movie is included in references. When the user types in all their file names and presses enter, the back end functions begin to run and return a verdict: match or mismatch. Based on this answer, the app will produce a popup that says “Match” or “Mismatch,” similar to what happens in the movie. The user can then reenter different files or close the program.

**Component Description Part II (Back End):**

As for the components within our code, we had a base class for all Clothing objects, and 5 derived classes: headwear, tops, jackets, bottoms, and footwear which derived its members. The derived classes inherited one main function—setColor(). This sets the color attribute of every clothing object to the correct value based on the average RGB values of the image.

In order to compute the average RGB values of the images for each derived class, we read in a ppm filename for each derived class and used a function to isolate the RGB values (0-255) of every pixel into an array. From this, we calculated the average values of this array for each respective RGB array.

Going back to the functionality of the setColor() function, the RGB values are obtained as previously mentioned. Based on these three average RGB values and a set of a predefined range of numbers within 0-255, we determined the overarching color of the image and set this to the inherited color attribute of the object.

After each object has been assigned a color, the isCompatible() function returns a boolean value based on whether the colors go together or not. For now, this function simply analyzes the top and bottom objects to determine whether the outfit is compatible. However, if we had more time, we would aim to analyze all 5 components of an outfit to determine this compatibility. We then pass this boolean result into the main function which tells the user whether the outfit is cute or not.

**Challenges & Decisions Made:**

The hardest aspect of our project, and where the vast majority of our time was dedicated, was figuring out how to read in data from images. We did not know how to use C++ to access each pixel of, let’s say a .JPEG, which is what we expect most users to have their images saved as. What we figured out was how to access the data for each pixel of a .PPM file, but our code only works for “p3” or “simple” images. With time restrictions, we were limited in what we were actually able to produce.

After writing the initial code to read in p3 images, we spent a lot of time trying different methods of reading in pixel data of images, since we did not want our project to be confined to p3 ppm files since that is not realistically how most images exist/what users have on hand. We ran into modules like ImageMagick++ and OpenCV. Many of the image processing algorithms we stumbled upon utilized this, and we thought it would be a perfect match for what we intended. However, we ran into issues with downloading the GitHub directories on Mac devices as well as obtaining the correct files to use OpenCV. Once we did get this to work, we felt a large learning curve with the OpenCV file and tried *numerous* different dominant color image analyses but none gave us success, even with simple image types.

As such, we decided to rely on the code that analyzed p3 files and use this as an example for how we wanted our code to actually work. By doing this, we were able to actually test how the rest of our files interacted with one another and whether we could actually produce a match or mismatch output, which we did end up finding success with.

We also really wanted to include some form of texture analysis, since we felt that being able to read what patterns clothes are and seeing if they match, was also a key part of our project. Since figuring out the color data was prerequisite to identifying patterns between the different colors, we ultimately did not have time to figure out how to do texture analysis. However, time went into looking for other source code and playing with it to see if it was a possibility.

We also were unsure if we would be able to make an application interface for our code given the time constraints. With image processing at the forefront of our minds, we knew our interface would not be anything spectacular. We eventually decided on using the Dear ImGui library to make a simple yet effective windows application to host our program. Navigating the Dear ImGui functions was a learning curve since it is not written in traditional C++ code that we see in class. After some getting used to, the Dear ImGui library saved us a lot of time and worked wonders to create a clean and bright interface.

**Core Algorithms/Routines:**

The algorithm used to retrieve the RGB values from a ppm image type essentially started by first opening an image. Since the images are P3 ppm files, the code just reads through the file and puts the RGB values into three separate strings, then converts these values from string values to integers to match the correct range from 0-255.

In our average value calculations, an algorithm we utilized was looping through the array to sum the total RGB pixel values (respectively), then find their average.

**References:**

1. ImGui repository: <https://github.com/ocornut/imgui.git>
2. ImGui Graphic: <https://github.com/ocornut/imgui/issues/3606#issuecomment-731146287>
3. ImGui User Tutorial: <https://youtu.be/vWXrFetSH8w>
4. Image of Clueless Computer Interface Inspiration:
5. RGB corresponding colors: <https://www.justfreetools.com/en/rgb-color>
6. Sample reading in color data code:
7. Video on how to read in color data: <https://www.youtube.com/watch?v=HGHbcRscFsg>
8. OpenCV how to read image data: <https://learnopencv.com/read-an-image-in-opencv-python-cpp/>
9. OpenCV How to use video:
10. ImageMagick++:
11. A texture Github repository we considered: <https://github.com/ASTex-ICube/semiproctex>