iCollab: A Collaborative Photo Editing Application

David Castaneda
Disha Patel
CNT4173

Introduction

The final project for this class required for the development of an open application that allows users to interact in a multi-user, graphical environment using native sockets. Inspired from Adobe Cloud's image manipulation applications such as Lightroom and Photoshop, iCollab is a graphical user interface that allows for up to ten users edit different aspects of a shared image. iCollab is set up so that users may communicate at the transport layer of the OSI model. It consists of different sliders that users control to manipulate an image that is updated in real time for all users connected to the server.

Methodology

In order to be able to communicate at a lower layer of communication the application needed to be designed using a language that allowed for socket programming. Python is a language that allows for socket programming, and GUI implementation, which is necessary for a photo editing application. Python also consists of many helpful libraries that we were able to implement into our application. These include: socket, threading, tkinter, PIL, etc. The socket library was useful for easily setting up the sockets used to connect clients and servers. The threading library is crucial to our application as it helped us set up an environment in which multiple users could manipulate the image simultaneously. The tkinter library allowed for us to develop a GUI in which users could interact, and see changes being made in a user-friendly manner. PIL is a very popular python imaging library that added image processing capabilities to our application.

The first step we took towards developing our application is designing the GUI and setting up functioning image editing capabilities. The libraries we used included tkinter and PIL.

While very helpful, the libraries needed to be carefully studied in order to be used correctly. Tkinter gives the developer much control on how they can set up the environment, also making it more complex. Initially the application was set up to be a specific size and as a result was not compatible with different screen resolutions. The application now is able to resize itself according to the resolution and appearance of the client's screen. Along with building the GUI, the PIL functions we used were also reconfigured to work more appropriately with our application. For example, the scale for brightness is set up from 0 to 200. We also found it more efficient to set up the image as an array of pixels to update each time the user made a change, instead of updating the image directly.

The next step was to set up our server and client system using native sockets. The sockets used exist on the TCP. Through our research we found it interesting that while a client only requires the function process of socket() then connect(), a server requires the following sequence: socket(), bind(), listen(), accept(). To allow for multiple users to connect to the server simultaneously we used multithreading. Each user was placed on a different thread that the server was listening on.

After that we needed to manipulate our already existing GUI to create an environment in which multiple clients could interact. Each option we had developed for manipulation was also put on a different thread so that multiple users could make changes at the same time. Calling lambda functions also made our code more efficient, and clean. We used lambda functions to set up our sliders in the GUI for users to change the image. The use of lambdas is mainly to set up callback functions. The lambda functions in our code are set up to return a function that changes the slider value.

Results

First the server is initiated using the terminal by typing the following command into their terminal: “python3 Server.py –H localhost –P 10000” (for running local, the host can be localhost). The server has been set up to be constantly listening for clients that request to connect. It prints status of clients as they connect and disconnect in the terminal. It also prints the total number of connections as this number gets updated.

For a client to connect to the application they must type “python3 Client.py –H localhost –P 1000” (host and port respective to where the server will be running). A window will open on the client’s screen that will have an already uploaded image opened up for users to modify. The application allows users to manipulate the following setting of a function:

- Brightness: scaled from 0 to 200. The slider original state is modified to be 100 marking the original brightness of the image.
- Contrast: scaled from 0 to 100. The slider is set to 0 by default.
- Hue: scaled from 0 to 100. The slider is set to 0 by default.
- Saturation: scaled from 0 to 500. The slider original state is modified to be 100 marking the original saturation of the image.
- Exposure: scaled from 0 to 100. The slider is set to 0 by default.
- Brilliance: scaled from 0 to 100. The slider is set to 0 by default.
- Sharpness: scaled from 0 to 200. The slider original state is modified to be 100 marking the original sharpness of image. Doing this allows user to both reduce and increase sharpness level.

Analysis

This project was a challenging but rewarding learning experience. Using native sockets instead of HTTP allowed for us to gain a deep understanding of how it is that a server allows for two clients to communicate with one another. While establishing a connection may seem easy, the process of sharing changes that clients make to an application in real time with other clients is a more difficult process. A user may simply move a slider with their keyboard, but that value is then converted and applied to different complex structures (such as an array), which is then in turn sent to other clients. This data is then, once again, restructured to be compatible, and presented on the GUI.

The iCollab application is a program that has much potential for enhancement. For example, the GUI's appearance can be further enhanced. Just as applications like Photoshop take advantage of the infinity edge for their menus/tools, the options for image manipulation can also

be rearranged to the infinity edge. Another enhancement may include allowing users to lock sliders so that other clients may not manipulate them if they are already in use. Another feature that can be added is the ability for users to have different administration rights. For example a standard user may only be able to update an already uploaded image, whereas a user with administrator rights may be able to upload new images to be edited.