**AggieSnap 2014**

**Team 62: “All We do is Nguyen”**

Nguyen Le Huynh

Shaeeta Sharar

Timothy Duus

**Work Distribution:**

Our team began by distributing small pieces of the program to individuals and then combining each element into a final working product. Most of the GUI setup was done by Timothy. He set up the window with all its individual widgets and base callbacks. He also worked on inputting the files. Shaeeta worked on the picture class. She set up operators and wrote the picture information to a file. She also worked on reading in from files, helped with inputting files and editing callbacks and improving the layout of the main GUI front. Nguyen worked on the tags class which included sorting the tags into their individual categories: the base of searching by tags. He coded the browse and search by tag functions. He, along with Shaeeta, combined all individual parts that were given to them. Nguyen also coded two optional extra credit problems. The report was written by and edited by Timothy and Shaeeta.

Errors were generally worked out individually as they were encountered. When faced with a particularly nasty error we often worked together to resolve it. Meetings were held weekly, both during and outside of class. During class we tended to work on our individual parts. When we met outside of class, we often worked on parts together.

**The Problem:**

“AggieSnap” called for us to design a picture database with a graphical user interface. The project included: getting a picture from a filename and the Web, adding tags, saving pictures, and browsing the pictures. The significance is it forces us to combine teamwork and programming knowledge to achieve a working design that meets specifications as one would in industry.

**Restrictions:**

A restriction we faced would be the graphics library we were required to use. Many of the individual parts of FLTK appear to behave similarly but in fact are very different. We encountered problems with this when we discovered that the attach() function is defined differently for Buttons and Images. This lead to problems in our program as we were mistakenly trying to attach an Image file after the reference had been deleted due to scope causing a segmentation fault. Also, due to the fact that we were using an abbreviated FLTK, many of the online resources were of no help to us.

Limitations to our program mostly reside in our reliance on proper user inputs. If the user is uploading a file from their directory, they must type the name and type exactly as it appears. We have a function that checks to make sure the type is correct, but not one to check if the file name is. Likewise, if the user uploads from the web they must name their file with the same type as the one they found online.

On a similar note, the user has to type our five tags in with capitalization and correct spelling. Although tag order doesn’t matter, if the user inputs “pets” or “Pet” instead of “Pets” there will be an error. Not accounting for this error made our program easier to code, but could lead to problems if the user doesn’t follow our instructions.

Another limitation that arose when searching by tags was searching without inputting any tags. This will cause the program to quit instead of displaying an error.

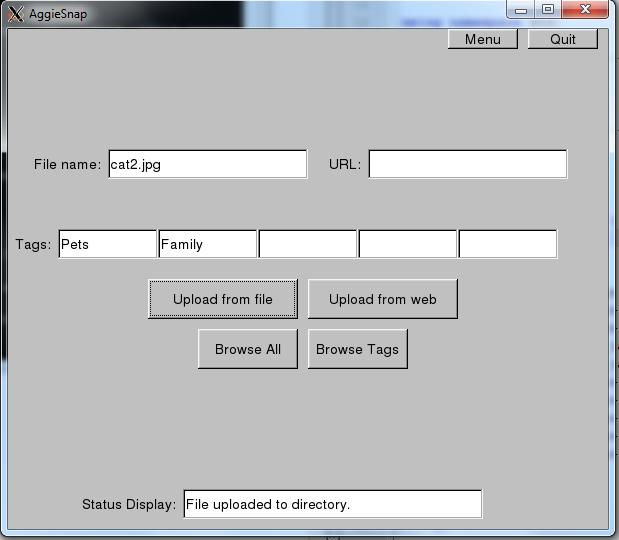
After uploading files or searching by tags, the user must manually clear all input boxes. This leads to a more tedious user interface. There aren’t any program errors that occur with this as having a filename typed in the input box doesn’t affect searching by tags. Also, after clicking the menu button to begin the program the user can’t return to the initial start page with all the instructions. This could cause an annoyance if the person wants to go back and see the instructions. These are limitations as they hinder user-friendliness.

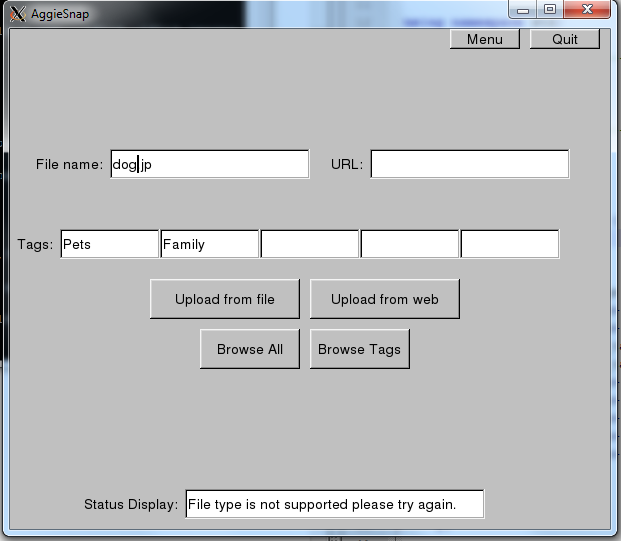
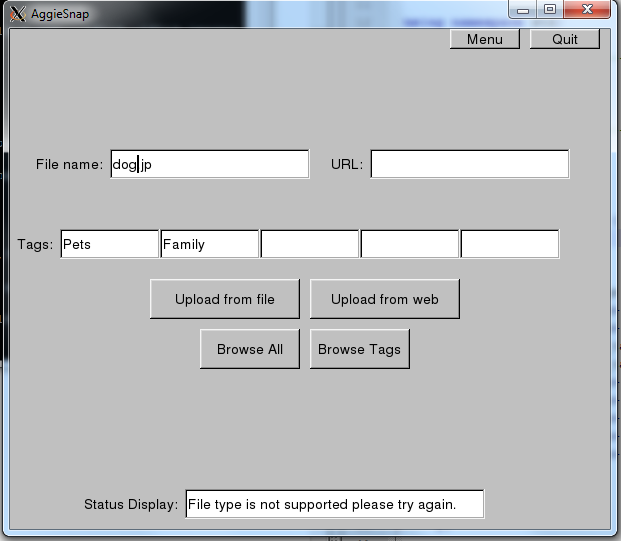
**Our Approach:**

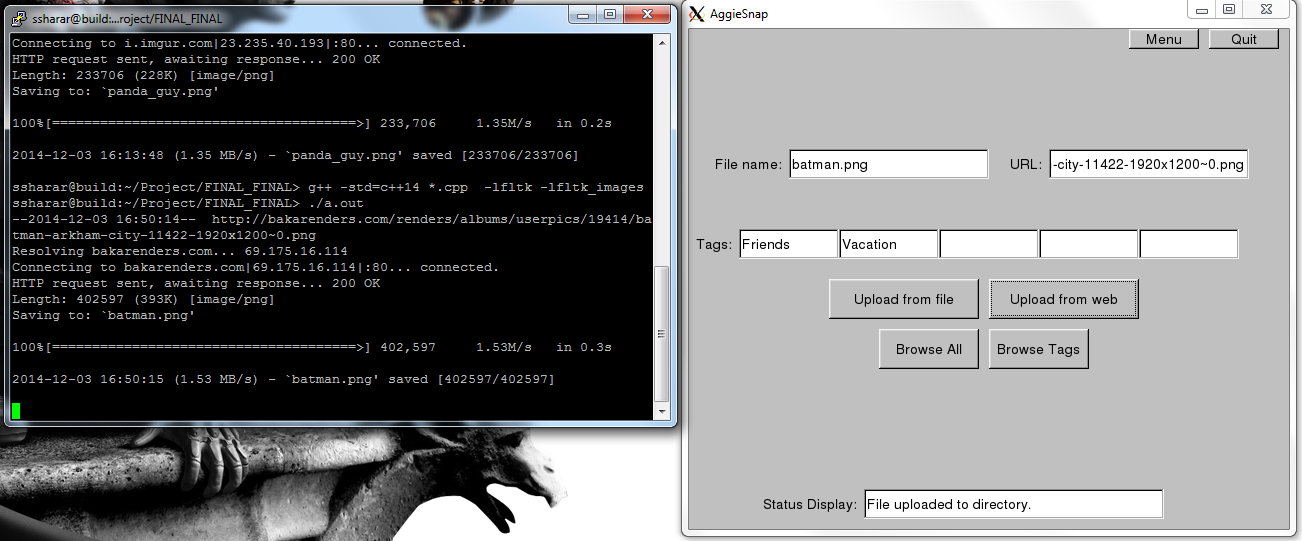
Our approach was to tackle the small problems and then combine individual parts to have a completed design. We first started off by writing the classes to help organize the back end of our program. Our classes were somewhat based on the Date class from the textbook. We used these to organize our images and tags. The picture class we created focused on reading and writing to the index file as well as sorting our main picture vector which held the image name and tags. The tags class organized the tags inputted and separated them into categories to be used for searching.

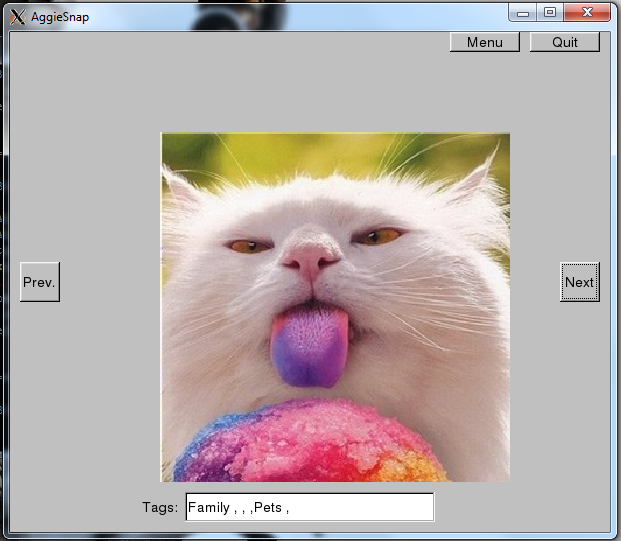
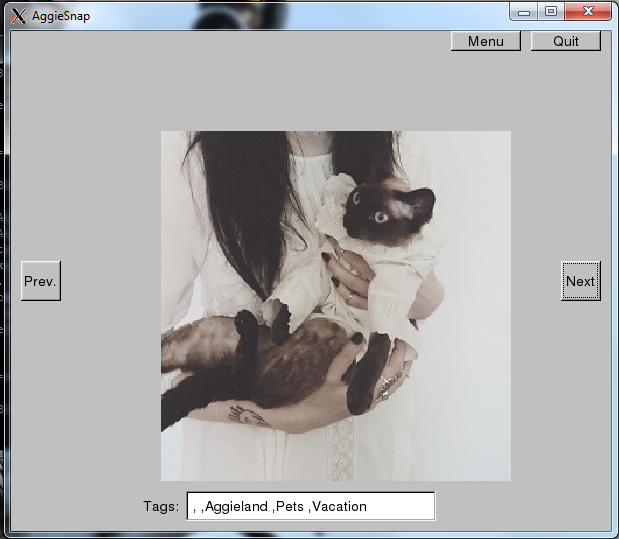
We used the Simple Window files of FLTK as a rough guide for our own window. This was a subclass of window that we created ourselves. We wrote the functions and GUI separately to make sure the logic in the functions was correct before attempting to mesh the two together. Most of the functions had to be tweaked a little to work with the GUI. The GUI is based off of one window and is designed to hide and show objects whenever the widgets are used.

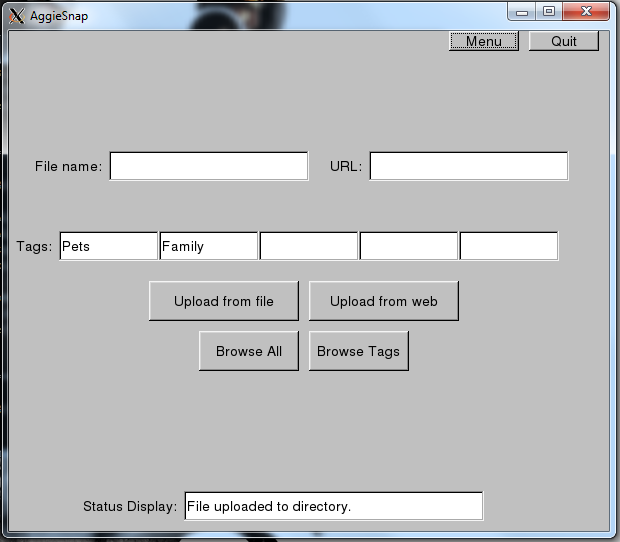
**Sample Run:**











**Results and Analysis:**

The program was designed by breaking it into smaller components to make it easier to debug and reduce the amount of work we felt we were doing at one time. Once the small parts were working individually, we combined them all using our GUI window subclass. We encountered multiple problems when implementing the GUI. One major problem was displaying the image when browsing. Originally we had an error with the scope of our image. Fixing this lead to a series of problems in which we added one too many functions hoping it would alleviate the errors while actually causing more. The solution involved using a vector\_ref instead of a plain vector and the “redraw” function. Another problem was browsing based on the tags combination. This required some technically challenging techniques to accomplish. We had to go in and add more functions for checking tags as well as editing the basic logic of our search by tags function. In addition to these major problems, we also faced many minor problems that involved small details of implementation. Some of these included errors with reading in the file index, uploading from the web, difficulty with the scope of our variables and the like. It was difficult to have a comprehensive file towards the end of the project. We would share our progress periodically, but many times one teammate would be working on one version of a file and the other would be changing that same file. This lead to different versions of the same initial files, which we had to integrate together. Often times, this itself lead to minor errors because while integrating, it’s often easy to miss minor changes that cause major problems. After overcoming all of these challenges, we were able to implement a functional design and logical program.

**Conclusions:**

We were given the problem of designing a user-friendly graphical program for displaying images. There were many specifications to our project as well as limitations that hindered our progress. Each team member brought unique approaches to solve these problems that we encountered.  Our final program was able to do all of the initial required tasks of getting a picture from a file and the Web, adding tags, saving pictures, and browsing the all the pictures and by specific tags. From the final program, we were able to show that it is possible to take C++ code and FLTK graphics to make a picture index with a GUI. However, there are still a few errors with our search by tags function. Primarily, it fails to separate images by a specific tag if there is more than one tag requested. Other than this, our program runs smoothly and allows for a comfortable user interface.

**Future Research:**

To further improve our program, it would be better to make multiple windows for the different tasks, so the code is not so dense. Currently, we only have one window to hold all of the widgets. Also, our program needs to be more user-friendly. We have a button for each function, but it would be better if we could combine some of the buttons and have the computer determine the function it needs to perform based on the widget input.

To extend on our program, we could change our tag input to check boxes. This would eliminate user input error and also allow the program to run smoothly. In addition, our extended program could allow the user to find a photo from its name and even rename the photo. This will allow the user to quickly show a friend a picture if they know the filename. Also, the user can correct any spelling errors made at the time of uploading.

**Instructions to run our program:**

Compiling and Running the Program: Start by placing all of the necessary source files into one file. Then, enable “X11 forwarding” and log on to the “build.tamu.edu” server. Navigate to your file with all of the source code. Once there, type “g++ -std=c++14 \*.cpp –lfltk –lfltk\_images” into the command line. Next, you’ll need to make sure “XMING” is running, so search for the “XMING” program on your computer. Now, type “./a.out” into the command line to run the program.

Starting the Program: To start the program, just click the “main” button and this will tell the computer to display all of the operating buttons like “upload from file” and “browse all”. This will allow you to get to the main menu, which is where all of the navigation takes place.

Uploading a Picture: Uploading a photo requires that the photo you wish to upload have a URL name or the picture saved in your file. Then, enter the filename into the filename input box, which should include the file extension, and, if it is from the internet, enter the URL into the corresponding input box. In addition, add any of the five tags (“Family”, “Friends”, “Aggieland”, “Vacation”, or “Pets”) to the tags’ boxes located below the “URL” input box. Click the appropriate upload button, “Upload from File” or “Upload from Web,” to complete the process. You must manually clear the input boxes to upload more photos.

Browsing ALL Photos: To browse ALL of you photos, click the “Browse All” button and use the “Next” and “Previous” buttons that pop up to navigate from picture to picture. If you wish to return to the main menu, click the “main menu” button in the top right corner of the screen.

Browsing by TAGS: Browsing by tags limits your search to only those photos that have been tagged with that tag name. To do this, input one of the tag names into the tag search box. The search is case-sensitive, so enter the tag exactly as you would want to search for. For example, “Aggieland” and “aggieland” are two different tags. Then, click the “Browse by Tags” button. This will allow you to use the “Next” and “Previous” buttons to navigate through the pictures. If you wish to return to the main menu, click the “main menu” button in the top right corner of the screen.

Exiting the program: Click the “Quit” button in the top right corner at any time to exit the program.

**Bibliography:**

Fine, Benjamin. Personal interview. Nov. 2014.

Stroustrup, Bjarne. *Programming: Principles and Practice Using C*. Upper Saddle River, NJ: Addison-Wesley, 2014. Print.