PicView

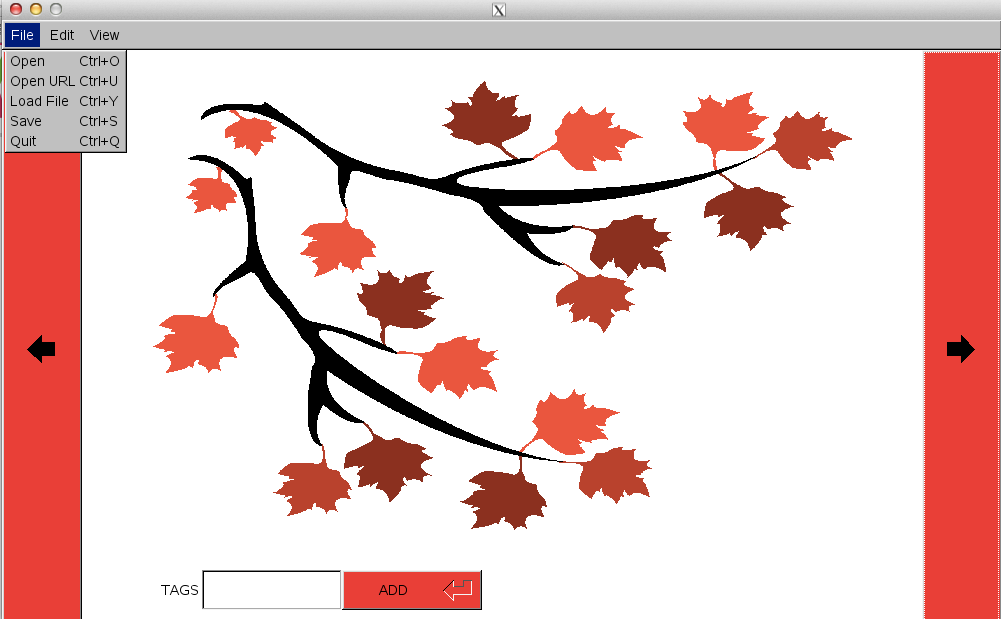
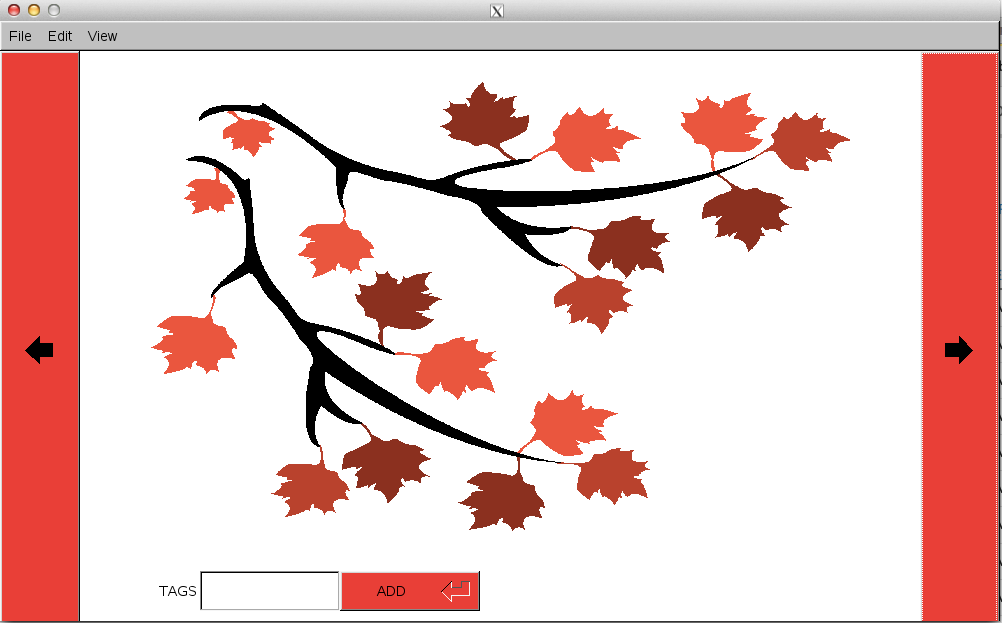
(#1) The program “PicView” was created using the creativity and skill sets of **Dawson Turechek, Wesley Vance, and Alexandra Vincent**. Dawson wrote the code for the function that searches through the vector for the pictures containing the tag the user is looking for. Wesley wrote the code for the way the program looks and opening pictures into the program. Alexandra wrote the code that adds tags to the specific picture in the database and figured out how buttons work so that the buttons can increment and decrement the vector to display the previous and next picture. The team as a whole believes each team member has contributed an equal amount and that each member did a fair share of the work.

(#2) The project that was assigned was to write a program that could open pictures from the user’s computer or from a URL that the user enters and save them to a database within the program that displays pictures, adds tags to pictures, and lets the user flip through the entire database or flip through pictures only containing a certain tag in a user friendly manner. This program is very useful because nowadays everyone has their pictures on their computer and adding tags to photos allows the user to narrow down their search through all their photos.

(#3) Throughout the process of designing and creating this program we didn’t come across many restrictions. The thing that probably restricted us the most was the time limit we were given to incorporate so many different features into this program. Another limitation was the FLTK graphics library. The GUI interface is also a limitation.

(#4) To approach this assignment we first thought about what we wanted the user to see. With this in mind, we designed the display first. We created the window, menu bar with submenus, and all the buttons. At this point the buttons didn’t do anything but everything the user would need to use the program was displayed. This code can be found in the file titled display.cpp. After we had all the code for this written we went into making windows for the options in the menu bar. The code for the menu bar can be found in the file titled menu\_bar.cpp. We also created header files for the display and the menu bar. These header files are display.h and menu\_bar.h. Once this was done we began working on the backend of the code. First we worked on getting the program to accept jpg, JPG, png, PNG, gif, GIF, bmp, BMP, pnm, and PNM. To do this we created a series of string checks and if statements. The code for this can be found in the file titled images.cpp. Next we created a vector that would be able to hold the names of the picture files that the user inputs and the tags that the user associates with each picture. Each element of the vector holds one file name for a picture and all the tags associated with that picture. This vector is our program’s database. All pictures and tags that are saved to the program are saved into this vector. The left and right arrow buttons can go through the elements of the vector. For example, if the user clicks the next button the program will display the next picture in the vector. The code for this can be found in the file titled database.cpp. We also created a header file for the database that is titled database.h. Throughout this process we continually added to the file titled Project.cpp. This file is where we bring the other cpp files together. This file includes all our header files and has a main function that handles the database function and displays all of our functions. At the very end we had issues making our database class visible to all source files but eventually with the help of google and a few emails back and forth with some helpful peer teachers we got it to work.

(#5)



(#6) Overall our program gave us very positive results. Throughout the whole process of designing and coding we kept the user in mind. Using the menu bar approach makes the program extremely user friendly because everything the user needs to navigate through the program is located in the menu bar. We also added code with the mindset that the code needed to be easy to edit and read for any programmer that looks at the code. We believe the code is well organized and allows for easy editing from any programmer that would want to work on the program.

(#7) This program was a great learning experience because it let us learn how our programming skills can be applied to the real world. Sometimes it’s hard to think about how a homework program that only uses one aspect of programming will help us in the real world but putting all those aspects together in one project really helps us see how we can use our programming knowledge outside of the classroom. Specifically from this project we learned how to open files from the computer and save them to our program. We also learned how to create shortcuts and get buttons to do what we want them to do.

(#8) The program works smoothly and in a user friendly manner but of course there are ways that the program could be improved. Our program could probably be improved by letting the user add more tags to the pictures. This would make the program even more user friendly because the user could describe their picture in more detail or different types of tag searches. Another way the program could be improved is by giving the user the option of displaying the pictures as thumbnails so the user could see more than one picture at a time. This would also improve the search function because when the user searches for a certain tag the user could see thumbnails of all the pictures pertaining to the tag they searched instead of having to continue clicking the next and previous button to see what pictures pertain to the tag the searched. Another way this program could be improved is by letting the user delete a picture or delete/edit a tag associated with a picture. Also, the GUI interface is not as modern looking so that could also be improved. We had tried to include the exif data but we simply ran out of time. If we had more time the exif data would have definitely been something we would have added to the program.

(#9) Once you open the program you can begin to add pictures to the database using the menu bar. Click File Open (Ctrl + O will also work). You may also open a picture using a URL from the internet (Ctrl + U will also work). Once you open the picture you can add tags by entering your tags into the box at the bottom of the window and save them to the picture by pressing the ADD button. You can save your picture and tags by going to File Save (Ctrl + S will also work). Tags can also be added or viewed by going to View Tags (Ctrl + T will also work). Repeat this process for each picture you want to add to the database. Searching through tags for pictures with specific tags can be done by going to Edit Search (Ctrl + C will also work). Using the right arrow will change the displayed picture to the next picture (Ctrl + D) and using the left arrow (Ctrl + A) will change the displayed picture to the previous picture. The description of what the program can do can be found by gong to View About (Ctrl + / will also work). To exit the program you can either press the red X in the top right corner of the window or you can go to File Exit (Ctrl + Q will also work).

(#10) Commented Code:

#include "header\_files/display.h"

#include "header\_files/menu\_bar.h"

int main() {

database.load("Database.txt"); //This should add two elements

display();

return(Fl::run());

}

//This is the code for the display of the application.

#include "header\_files/display.h"

#include "header\_files/menu\_bar.h"

Fl\_Window \*win;

static int database\_element=0;

void display(){

win = new Fl\_Window(200,100,1000,600); //Creates a new large window (Everything is inside this)

win->color(FL\_WHITE);

draw\_image(database\_element); //This is where we draw the picture in a "virtual box", and display its title/path. This is in images.cpp

menu(database\_element); //This creates the menu bar, its buttons and all its functions. this is in menu\_bar.cpp

//This will draw a button that will incrase the element in a vector that holds the paths to pictures. EX: Vec[0] = example1.jpg, Vec[1] = example2.jpg

Fl\_Button\* next\_pic = new Fl\_Button( 920, 30, 80, 600, "@+9->" );

next\_pic-> shortcut(FL\_CTRL + 'd');

next\_pic-> color(FL\_RED);

next\_pic-> callback( ( Fl\_Callback\* ) next\_pic\_cb );

//this will draw a button that will decrease the element in a vector which will then redraw the picture of the new path.

Fl\_Button\* previous\_pic = new Fl\_Button( 0, 30, 80, 600, "@+9<-" );

previous\_pic -> shortcut(FL\_CTRL + 'a');

previous\_pic -> color(FL\_RED);

previous\_pic -> callback( ( Fl\_Callback\* ) previous\_pic\_cb );

//This is where the inputs for tags will be inputed.

Fl\_Input\* tag\_input = new Fl\_Input( 200, 550, 140, 40, "TAGS" );

Fl\_Button\* add\_tag = new Fl\_Return\_Button(340,550,140,40, "ADD" );

add\_tag -> color(FL\_RED);

add\_tag->callback(add\_tag\_cb, tag\_input);

win->end();

win->show();

}

void next\_pic\_cb( Fl\_Widget\* obj , void\* )

{

obj->label( "@+9->" );

obj->redraw();

database.next\_pic(database\_element);

draw\_image(database\_element);

win->redraw();

}

void previous\_pic\_cb( Fl\_Widget\* obj , void\* )

{

obj->label( "@+9<-" );

obj->redraw();

database.previous\_pic(database\_element);

draw\_image( database\_element);

win->redraw();

}

void add\_tag\_cb(Fl\_Widget\* obj, void\* inp)

{

obj->label("ADD");

obj->resize(340,550,140,40);

obj->redraw();

string input;

input=static\_cast<Fl\_Input\*>(inp)->value();

database.add\_tag(database\_element, input);

cout<<database.element(database\_element)<<endl;

}

#include "header\_files/display.h"

#include "header\_files/menu\_bar.h"

Fl\_JPEG\_Image\* my\_jpeg\_image;

Fl\_BMP\_Image\* my\_bmp\_image;

Fl\_PNG\_Image\* my\_png\_image;

Fl\_PNM\_Image\* my\_pnm\_image;

Fl\_GIF\_Image\* my\_gif\_image;

Fl\_Box \*box;

void draw\_image(int database\_element){

const char\* file\_name;//This gets the path out of the elemenet

string start\_file="";

string current=database.element(database\_element);

for(int i=0; i<current.size();i++){

if (current[i]!=','){

start\_file+=current[i];

}

else {

break;

}

}

file\_name=start\_file.c\_str();

const char\* error\_file;

error\_file= "pictures/error\_file.png";

box = new Fl\_Box(150,60,700,465,"");

string file= file\_name;

string jpeg\_check(".jpg");

string jpeg\_check2(".JPG");

string jpeg\_check3(".jpeg");

string jpeg\_check4(".JPEG");

string png\_check(".png");

string png\_check2(".PNG");

string gif\_check(".gif");

string gif\_check2(".GIF");

string bmp\_check(".bmp");

string bmp\_check2(".BMP");

string pnm\_check(".pnm");

string pnm\_check2(".PNM");

if (file.find(jpeg\_check) != string::npos){

cout<<"You found a .jpg "<<file\_name<<endl;

my\_jpeg\_image = new Fl\_JPEG\_Image(file\_name);

Fl\_Image \*scaled\_image = my\_jpeg\_image->copy(700, 450);

box->image(scaled\_image);

}

else if (file.find(jpeg\_check2) != string::npos){

cout<<"You found a .JPG "<<file\_name<<endl;

my\_jpeg\_image = new Fl\_JPEG\_Image(file\_name);

Fl\_Image \*scaled\_image = my\_jpeg\_image->copy(700, 450);

box->image(scaled\_image);

}

else if (file.find(jpeg\_check3) != string::npos){

cout<<"You found a .jpeg "<<file\_name<<endl;

my\_jpeg\_image = new Fl\_JPEG\_Image(file\_name);

Fl\_Image \*scaled\_image = my\_jpeg\_image->copy(700, 450);

box->image(scaled\_image);

}

else if (file.find(jpeg\_check) != string::npos){

cout<<"You found a .JPEG "<<file\_name<<endl;

my\_jpeg\_image = new Fl\_JPEG\_Image(file\_name);

Fl\_Image \*scaled\_image = my\_jpeg\_image->copy(700, 450);

box->image(scaled\_image);

}

else if (file.find(png\_check) != string::npos){

cout<<"You found a .png "<<file\_name<<endl;

my\_png\_image =new Fl\_PNG\_Image(file\_name);

Fl\_Image \*scaled\_image = my\_png\_image->copy(700, 450);

box->image(scaled\_image);

}

else if (file.find(png\_check2) != string::npos){

cout<<"You found a .PNG "<<file\_name<<endl;

my\_png\_image =new Fl\_PNG\_Image(file\_name);

Fl\_Image \*scaled\_image = my\_png\_image->copy(700, 450);

box->image(scaled\_image);

}

else if (file.find(gif\_check) != string::npos){

cout<<"You found a .gif "<<file\_name<<endl;

my\_gif\_image= new Fl\_GIF\_Image(file\_name);

Fl\_Image \*scaled\_image = my\_gif\_image->copy(700, 450);

box->image(scaled\_image);

}

else if (file.find(gif\_check2) != string::npos){

cout<<"You found a .GIF "<<file\_name<<endl;

my\_gif\_image= new Fl\_GIF\_Image(file\_name);

Fl\_Image \*scaled\_image = my\_gif\_image->copy(700, 450);

box->image(scaled\_image);

}

else if (file.find(bmp\_check) != string::npos){

cout<<"You found a .bmp "<<file\_name<<endl;

my\_bmp\_image= new Fl\_BMP\_Image(file\_name);

Fl\_Image \*scaled\_image = my\_bmp\_image->copy(700, 450);

box->image(scaled\_image);

}

else if (file.find(bmp\_check2) != string::npos){

cout<<"You found a .BMP "<<file\_name<<endl;

my\_bmp\_image= new Fl\_BMP\_Image(file\_name);

Fl\_Image \*scaled\_image = my\_bmp\_image->copy(700, 450);

box->image(scaled\_image);

}

else if (file.find(pnm\_check)!= string::npos){

cout<<"You found a .pnm "<<file\_name<<endl;

my\_pnm\_image= new Fl\_PNM\_Image(file\_name);

Fl\_Image \*scaled\_image = my\_pnm\_image->copy(700, 450);

box->image(scaled\_image);

}

else if (file.find(pnm\_check2)!= string::npos){

cout<<"You found a .PNM "<<file\_name<<endl;

my\_pnm\_image= new Fl\_PNM\_Image(file\_name);

Fl\_Image \*scaled\_image = my\_pnm\_image->copy(700, 450);

box->image(scaled\_image);

}

else {

cout<<"You did not load an appropriate image..."<<endl;

my\_png\_image = new Fl\_PNG\_Image(error\_file);

Fl\_Image \*scaled\_image = my\_png\_image->copy(700, 450);

box->image(scaled\_image);

}

}

//This code is dedicated to the menu bar ontop of the application.

#include "header\_files/display.h"

#include "header\_files/menu\_bar.h"

void menu(int database\_element){

Fl\_Menu\_Bar \*menu = new Fl\_Menu\_Bar(0,0,1000,30);

menu->add("File/Open", FL\_CTRL+'o', Open\_CB);

menu->add("File/Open URL", FL\_CTRL+'u', Open\_URL\_CB);

menu->add("File/Load File",FL\_CTRL+'y', Load\_File\_CB);

menu->add("File/Save", FL\_CTRL+'s', Save\_CB);

menu->add("File/Quit", FL\_CTRL+'q', Quit\_CB);

menu->add("Edit/Search", FL\_CTRL+'c', Search\_CB);

menu->add("View/Tags", FL\_CTRL+'t', Tags\_CB);

//menu->add("View/Exif", FL\_CTRL+'e', Exif\_CB);

menu->add("View/About", FL\_CTRL+'/', About\_CB);

}

//---------------THIS IS TAKEN FROM FLTK DOCUMENTATION-------------------

//-----LOADS A FILE CHOOSER AND ALLOWS USER TO PICK A FILE---------------

void Load\_File\_CB(Fl\_Widget\*, void\*){

Fl\_Native\_File\_Chooser file\_chooser;

file\_chooser.title("Pick a Saved File/ Database");

file\_chooser.type(Fl\_Native\_File\_Chooser::BROWSE\_FILE);

file\_chooser.filter("Save Files\t\*.{txt}\n");

file\_chooser.directory("/home/ugrads/w/wvance/"); // default directory to use

switch ( file\_chooser.show() ) {// Show native chooser

case -1: cout<<"ERROR " <<file\_chooser.errmsg()<<endl; break; // ERROR

case 1: cout<<"CANCEL" <<endl; break; // CANCEL

default: cout<<file\_chooser.filename() <<endl; break; // FILE CHOSEN

}

string load\_file=file\_chooser.filename();

database.load(load\_file);

}

//------------------------------------------------------------------------

//---------THIS FUNCTION IS USED TO ADD A NEW PICTURE TO THE DATABASE-----

void Open\_CB(Fl\_Widget\*, void\*) {

Fl\_Native\_File\_Chooser file\_chooser;

file\_chooser.title("Pick a Photo");

file\_chooser.type(Fl\_Native\_File\_Chooser::BROWSE\_FILE);

file\_chooser.filter("Photos\t\*.{jpg,jpeg,gif,bmp,png}\n" "Program Files\t\*.{cxx,h,c,cpp}");

file\_chooser.directory("/home/ugrads/w/wvance"); // default directory to use

switch ( file\_chooser.show() ) {// Show native chooser

case -1: cout<<"ERROR " <<file\_chooser.errmsg()<<endl; break; // ERROR

case 1: cout<<"CANCEL" <<endl; break; // CANCEL

default: cout<<file\_chooser.filename() <<endl; break; // FILE CHOSEN

}

string open\_file= file\_chooser.filename();

database.add\_pic(open\_file);

}

//------------------------------------------------------------------------

//---------THIS OPENS A WINDOW THAT DISPLAYS THE TAGS IN THE VECTOR-------

void Tags\_CB(Fl\_Widget \*w, void \*) {

Fl\_Window \*win = new Fl\_Window(500, 300,450,200);

Fl\_Text\_Buffer \*buff = new Fl\_Text\_Buffer();

Fl\_Text\_Display \*disp = new Fl\_Text\_Display(20, 20, 450-40, 200-75, "Picture Tags");

disp ->buffer(buff);

win ->resizable(\*disp);

win ->show();

string output;

output = database.display();

buff->text(output.c\_str()); //Just display entire text file if needed

}

//------------------------------------------------------------------------

//---------THIS OPENS A WINDOW TO ALLOW USER TO INPUT A URL TO GRAB-------

void Open\_URL\_CB(Fl\_Widget \*w, void \*){

Fl\_Window \*win = new Fl\_Window(500, 300,450,200);

Fl\_Input\* URL\_input = new Fl\_Input ( 100, 110, 220, 40, "URL Picture" );

Fl\_Button\* add\_URL = new Fl\_Return\_Button (320,110,110,40, "ADD" );

add\_URL -> color (FL\_RED);

add\_URL -> callback(add\_URL\_CB, URL\_input);

Fl\_Text\_Buffer \*buff = new Fl\_Text\_Buffer ();

Fl\_Text\_Display \*disp = new Fl\_Text\_Display (20, 20, 450-40, 200-130, "URL Grabber");

disp ->buffer(buff);

win ->resizable(\*disp);

win ->show();

buff ->text("\n" " Please Place your URL Input Below: \n");

}

void add\_URL\_CB(Fl\_Widget \*w, void\* inp){//This will take the URL in the buffer and do stuff to it. hw3pr3

string input\_url;

input\_url=static\_cast<Fl\_Input\*>(inp)->value();

string file\_type;

if (input\_url.find(".jpeg")){

file\_type="pictures/download.jpeg";

}

else if (input\_url.find(".bmp")){

file\_type="pictures/download.bmp";

}

else if (input\_url.find(".gif")){

file\_type="pictures/download.gif";

}

else if (input\_url.find("pnm")){

file\_type="pictures/download.pnm";

}

else if (input\_url.find(".png")){

file\_type="pictures/download.png";

}

else if (input\_url.find(".png")){

file\_type="pictures/download.png";

}

else (cout<<"You did not enter a valid picture URL");

string command = "wget "+ input\_url +" -O"+ file\_type;

system(command.c\_str());

string save\_url\_path;

database.add\_pic(file\_type);

}

//------------------------------------------------------------------------

//----------SEARCHES THE TAGS WITHIN THE VECTOR---------------------------

void Search\_CB(Fl\_Widget \*w, void \*) {

}

//------------------------------------------------------------------------

//-------------QUITS THE PROGRAM ASKS USER TO SAVE------------------------

void Quit\_CB(Fl\_Widget \*, void \*) {

Fl\_Window \*win = new Fl\_Window(500, 300,450,200,"WOULD YOU LIKE TO SAVE?");

Fl\_Button\* YES = new Fl\_Button( 100, 50, 80, 70, "YES" );

YES -> color (FL\_RED);

YES -> callback( ( Fl\_Callback\* ) YES\_CB );

Fl\_Button\* NO = new Fl\_Button( 250, 50, 80, 70, "NO" );

NO -> color (FL\_RED);

NO -> callback( ( Fl\_Callback\* ) NO\_CB );

win -> color(FL\_WHITE);

win ->show();

}

void YES\_CB(Fl\_Widget \*w, void \*) {

database.save();

cout<<"You Saved Your File:"<<endl;

exit(0);

}

void NO\_CB(Fl\_Widget \*w, void \*) {

exit(0);

}

//------------------------------------------------------------------------

//---------SAVES THE PROGRAM TO A TEXTFILE WHICH CAN BE OPENED LATER------

void Save\_CB(Fl\_Widget \*, void \*) {

database.save();

}

//------------------------------------------------------------------------

//------------THIS IS AN ADVANCED FEATURE THAT DOESNT WORK YET------------

void Exif\_CB(Fl\_Widget \*, void \*) {

Fl\_Window \*win = new Fl\_Window(500,300,400,150);

win->end();

win->show();

}

//------------------------------------------------------------------------

//------------------ABOUT ME WINDOW WITH WHAT PROGRAM DOES----------------

void About\_CB(Fl\_Widget\*, void\*) {

Fl\_Window \*win = new Fl\_Window(500, 300,450,200);

Fl\_Text\_Buffer \*buff = new Fl\_Text\_Buffer();

Fl\_Text\_Display \*disp = new Fl\_Text\_Display(20, 20, 450-40, 200-40, "About PicView");

disp ->buffer(buff);

win ->resizable(\*disp);

win ->show();

buff->text("\n"

" This program 'PicView' stores and displays an image. \n"

" These images are a part of a database, saved with tags. \n"

" Use the menu bar above to access additional features. \n"

" --------------------------------------------------------------------------- \n"

" Picview was created as a project at TAMU for CSCE 121. \n"

" Credit for this project goes to: \n"

" Wesley Vance, Alex Vincent, Dawson Turechek \n" );

}

//------------------------------------------------------------------------

//------------------------------------------------------------------------

#include "header\_files/display.h"

#include "header\_files/menu\_bar.h"

//database.cpp

Database database;

Database::Database(){//Default database savefile to load.

}

Database::~Database(){

}

int Database::save()//This should open a textfile, and save the contents of the vector into the textfile.

{

ofstream save\_data;

if (!save\_data){

cout << "File couldn't be opened." << endl;

return -1;

}

save\_data.open("Database.txt");

for (int i=0; i< database\_vector.size(); i++){

save\_data<<database\_vector[i]<<"\n";

}

save\_data.close();

return 1;

}

int Database::load(string textfile) //Loads a .txt file into a vector

{

string line;

ifstream load\_data;

if (!load\_data){

cout << "File couldn't be opened." << endl;

return -1;

}

load\_data.open(textfile);

while (getline(load\_data, line)){

database\_vector.push\_back(line);

}

load\_data.close();

return 1;

}

int Database::add\_pic(string path){ //This adds a picture to the vector in a new element.

database\_vector.push\_back(path);

return 1;

}

int Database::add\_tag(int element, string tag){ // This function adds a tag to the vector then later will save it to the database.txt file.

database\_vector[element] += ","; // Find element then add a ',' to indicate a new tag.

database\_vector[element]+= tag;

return 1;

}

int Database::next\_pic(int &current\_element){

if (current\_element != database\_vector.size()-1){ //Handles cases when you click next button and there isnt element

++current\_element;

}

else (cout<<"There are no more pictures"<<endl);

return current\_element;

}

int Database::previous\_pic(int &current\_element){

if (current\_element!=0){

--current\_element;

}

else(cout<<"There are no more pictures"<<endl);

return current\_element;

}

string Database::display(){

// while()

for (int i=0;i<database\_vector.size();i++)

{

string output;

output=database\_vector[i]+"\n";

return output;

}

}

string Database::display\_tags(int current\_element){

string display\_tags;

display\_tags= database\_vector[current\_element];

return display\_tags;

}

string Database::element(int x){//This returns the string of whatever is in the vector.

string element;

element = database\_vector[x];

return element;

}

(#11) Bibliography

<http://stackoverflow.com/questions/2393345/how-to-append-text-to-a-text-file-in-c>

<http://www.fltk.org/doc-1.1/basics.html>

<http://www.fltk.org/doc-2.0/html/string_8h.html>

<http://seriss.com/people/erco/fltk/>

<http://stackoverflow.com/questions/297822/global-instance-of-a-class-in-c>