#include "std\_lib\_facilities\_4.h"

#include "Simple\_window.h"

#include "Window.h"

#include "Graph.h"

#include "GUI.h"

using namespace Graph\_lib;

class Intro\_Window : Graph\_lib::Window{

//Data Members

Button continue\_button;

Text welcome;

Text instructions;

Text tag\_instructions;

Text next\_instructions;

static void cb\_continue(Address,Address);

void continue\_on();

public:

Intro\_Window(Point xy, int w, int h, const string& title);

};

struct Tag\_obj;

struct Pic\_obj;

//Class Display\_Window creates the window and allows one to take input, manages the buttons and displays images

class Display\_Window : Graph\_lib::Window{

// Data members

ifstream ifs;

ofstream ofs;

vector<string> index\_read; //used in next()

string temp\_str; //also used in next()

//string next\_image;

// Constants

int index = 0; // An index of the current db line number

bool search\_mode; // true if the program is in search mode

string db\_filename = "db.txt";

vector<string> tag\_buttons\_pressed;// Use to save or search for Pic\_obj's

// Buttons

Button next\_button; //click to view the next picture

Button previous\_button; //click to view previous picture

Button home\_button; //click to return to viewing all images

Button family;

Button friends;

Button aggieland;

Button pets;

Button vacation;

Button search\_button;

Button add\_file\_button;

Button quit\_button;//click to exit program

In\_box input\_file;//allows user to input a new file name and tags

In\_box input\_url;

Out\_box mode; //indicates what images are being displayed

Out\_box tags\_displayed; //displays what tags are showing

Image \*p;

// Function Declarations

static void cb\_next(Address,Address);

void next();

static void cb\_previous(Address,Address);

void previous();

static void cb\_home(Address,Address);

void home();

static void cb\_family(Address,Address);

void tag0();

static void cb\_friends(Address,Address);

void tag1();

static void cb\_aggieland(Address,Address);

void tag2();

static void cb\_pets(Address,Address);

void tag3();

static void cb\_vacation(Address,Address);

void tag4();

static void cb\_search(Address,Address);

void search();

static void cb\_add\_file(Address,Address);

void add\_file();

static void cb\_quit(Address,Address);

void quit();

/\*static void cb\_search(Address,Address);

void find();

static void cd\_add(Address, Address);

void add();\*/

public:

Display\_Window(Point xy, int w, int h, const string& title);

vector<string> results;

vector<string> f\_search(string db\_fname, Tag\_obj t);

void check\_index\_range(); // Corrects index range errors

void set\_search\_mode(bool b); // Changes the search\_mode indicator and display

void draw\_image(string fname); // takes an image and draws it to the screen at Point(50,20)

void Display\_active\_tags();

int family\_i, friends\_i, aggieland\_i, pets\_i, vacation\_i; //tags

string family\_s, friends\_s, aggieland\_s, pets\_s, vacation\_s; //human-readable text that is written to index file

};

//Error message window

struct Error\_window : Graph\_lib::Window{

Error\_window(Point xy, int w, int h, const string& title );

private:

Text message;

Button ok\_button;

void ok();

static void cb\_ok(Address, Address);

};

// Tag\_obj is used in the search function

struct Tag\_obj

{

//contains tags

bool family;

bool friends;

bool aggieland;

bool pets;

bool vacation;

Tag\_obj()

{

family = false;

friends = false;

aggieland = false;

pets = false;

vacation = false;

}

Tag\_obj(bool dfam, bool dfr, bool dag, bool dpt, bool dva)

{

family = dfam;

friends = dfr;

aggieland = dag;

pets = dpt;

vacation = dva;

}

};

//Pic\_object class is used in the search function

struct Pic\_obj

{

string URLstring;

// Parts of a database object:

string file\_name;

Tag\_obj tags;

Pic\_obj()

{

file\_name = "db.txt";

Tag\_obj tags();

}

Pic\_obj(string dpname, Tag\_obj dt)

{

file\_name = dpname;

tags = dt;

}

};

// Handles reading and writing from the database---------------//

class db\_access

{

public:

vector<string> find\_tags; //find\_tags keeps track of which tags to search for

//string db\_filename = "Test\_db.txt";

ofstream ofs; // write to db

ifstream ifs; // read db

// Functions for taking input:

void check\_ftype(string file\_name); // Check type of file/if allowed

void check\_ltype(string location); // Decide if looking for a file location or URL location

void add\_picture(string location, string filename); // Download File/URL and name it filename

//- Allow recovery if Url can’t be downloaded/opened from disk

//- Allow recovery if incorrect file/url format

// Function for saving input:

void add\_tags(vector<string> tags\_entered); // tags entered to be added to the database

vector<string> search\_tags(); // tags entered in the search box

void record\_obj(string disk\_ad); // Check if db\_file exist, then save input to database text file

Pic\_obj load\_obj(int line\_to\_read); // creates a pic\_obj from a line in the db

void create\_db(string db\_name); // Creates database with file name db\_name (txt file)

};

//tags

// int family, friends, aggieland, pets, vacation;

**/////////////////////////////////////////////////////////////////////////**

**/Display.cpp////////////////////////////////////////////////////////////**

**/////////////////////////////////////////////////////////////////////////**

#include "universal.h"

Intro\_Window::Intro\_Window(Point xy, int w, int h, const string& title):Window(xy,w,h,title),

continue\_button(Point(x\_max()-440,480),70,20,"Continue",cb\_continue),

welcome(Point(x\_max()-650,80),"Welcome to AggieSnap!"),

instructions(Point(x\_max()-750,200),"Step 1: Click 'home' and upload pictures by putting the name of the image and/or URL."),

tag\_instructions(Point(x\_max()-750,220),"Step 2: Tag pictures by clicking all the tags that apply, then click 'Add' "),

next\_instructions(Point(x\_max()-750,240),"Step 3: Use the 'next' and 'previous' buttons to nagivate through pictures, search by clicking tags and then 'search' ")

{

welcome.set\_font\_size(50);

welcome.set\_color(Color::dark\_red);

attach(continue\_button);

attach(welcome);

attach(instructions);

attach(tag\_instructions);

attach(next\_instructions);

}

Display\_Window::Display\_Window(Point xy, int w, int h, const string& title):Window(xy,w,h,title),

quit\_button(Point(x\_max()-70,0),70,20,"Quit",cb\_quit),

next\_button(Point(x\_max()-70,30),70,20,"Next",cb\_next),

previous\_button(Point(x\_max()-70,60),70,20,"Previous",cb\_previous),

home\_button(Point(x\_max()-70,90),70,20,"Home",cb\_home),

family(Point(x\_max()-600,480),70,20,"Family",cb\_family),

friends(Point(x\_max()-520,480),70,20,"Friends",cb\_friends),

aggieland(Point(x\_max()-440,480),70,20,"Aggieland",cb\_aggieland),

pets(Point(x\_max()-360,480),70,20,"Pets",cb\_pets),

vacation(Point(x\_max()-280,480),70,20,"Vacation",cb\_vacation),

search\_button(Point(x\_max()-200,480),70,20,"Search",cb\_search),

add\_file\_button(Point(x\_max()-95,510),70,20,"Add",cb\_add\_file),

input\_file(Point(x\_max()-600,510),500,20,"Add File:"),

input\_url(Point(x\_max()-600,530),500,20,"Add URL"),

mode(Point(x\_max()-400,20),120,30,"Displaying: "),

tags\_displayed(Point(x\_max()-600,460),500,20,"Active Tags:")

{

attach(next\_button);

attach(previous\_button);

attach(home\_button);

attach(family);

attach(friends);

attach(aggieland);

attach(pets);

attach(vacation);

attach(search\_button);

attach(add\_file\_button);

attach(quit\_button);

attach(input\_file);

attach(input\_url);

attach(mode);

attach(tags\_displayed);

// Set all buttons to unpressed state

family\_i = 0;

friends\_i = 0;

aggieland\_i = 0;

pets\_i = 0;

vacation\_i = 0;

}

// Holla back Functions!-----------------------------

void Display\_Window:: cb\_quit(Address, Address pw)

{

reference\_to<Display\_Window>(pw).quit();

}

void Display\_Window:: cb\_next(Address,Address pw)

{

reference\_to<Display\_Window>(pw).next();

}

void Display\_Window:: cb\_previous(Address,Address pw)

{

reference\_to<Display\_Window>(pw).previous();

}

void Display\_Window:: cb\_home(Address,Address pw)

{

reference\_to<Display\_Window>(pw).home();

}

void Display\_Window:: cb\_family(Address,Address pw)

{

reference\_to<Display\_Window>(pw).tag0();

}

void Display\_Window:: cb\_friends(Address,Address pw)

{

reference\_to<Display\_Window>(pw).tag1();

}

void Display\_Window:: cb\_aggieland(Address,Address pw)

{

reference\_to<Display\_Window>(pw).tag2();

}

void Display\_Window:: cb\_pets(Address,Address pw)

{

reference\_to<Display\_Window>(pw).tag3();

}

void Display\_Window:: cb\_vacation(Address,Address pw)

{

reference\_to<Display\_Window>(pw).tag4();

}

void Display\_Window:: cb\_search(Address,Address pw)

{

reference\_to<Display\_Window>(pw).search();

}

void Display\_Window:: cb\_add\_file(Address,Address pw)

{

reference\_to<Display\_Window>(pw).add\_file();

}

void Intro\_Window:: cb\_continue(Address,Address pw)

{

reference\_to<Intro\_Window>(pw).continue\_on();

}

// Actual Functions ------------------------------------//

void Intro\_Window::continue\_on() //closes the intro window

{

hide();

}

void Display\_Window::set\_search\_mode(bool b)

{

//search\_mode = b;

//index = 0; // Always reset index when switching modes

}

// check\_index\_range Corrects index range errors //

// I might need to use qualified names for the vars...

void Display\_Window::check\_index\_range()

{

if (index < 0)

{

cerr << "Index ranger error.";

index = 0;

}

}

void Display\_Window::next()

{

string raw\_string,pic\_name;

int str\_start,str\_end;

if(search\_mode)

{

if(index < 0)

{

check\_index\_range();

}

else

{

draw\_image(results[index]);

index = index + 1;

}

}

else if(!search\_mode)

{

if(index < 0)

{

check\_index\_range();

}

else

{

index\_read.clear(); //Stores each line of db\_filename into vector index\_read. Then reads index\_read[index] and extracts file name and opens it.

ifs.open(db\_filename);

string tag\_aray[] = { "family", "friends", "aggieland", "pets", "vacation" };

while(!ifs.eof())

{

getline(ifs, raw\_string);

for (int i = 0; i <= 4; i++)

{

bool found\_t = raw\_string.find(tag\_aray[i]);

if (found\_t)

{

if ((i == 0) && (found\_t>0))

{

family\_i = 1;

}

if ((i == 1) && (found\_t>0))

{

friends\_i = 1;

}

if ((i == 2) && (found\_t>0))

{

aggieland\_i = 1;

}

if ((i == 3) && (found\_t>0))

{

pets\_i = 1;

}

if ((i == 4) && (found\_t>0))

{

vacation\_i = 1;

}

}

}

Display\_active\_tags();

str\_start = raw\_string.find('(') + 1; // gets position of the start of the pic\_name

str\_end = raw\_string.find(',') - 1; // gets position of the end of the pic\_name

pic\_name = raw\_string.substr(str\_start, str\_end);

index\_read.push\_back(pic\_name);

}

ifs.close();

draw\_image(index\_read[index]);

index = index + 1;

}

}

}

void Display\_Window::previous()

{

string raw\_string,pic\_name;

int str\_start,str\_end;

if(search\_mode)

{

if(index < 0)

{

check\_index\_range();

}

else

{

draw\_image(results[index]);

index = index - 1;

}

}

else if(!search\_mode)

{

if(index < 0)

{

check\_index\_range();

}

else

{

index = index - 1;

index\_read.clear(); //Stores each line of db\_filename into vector index\_read. Then reads index\_read[index] and extracts file name and opens it.

ifs.open(db\_filename);

while(!ifs.eof())

{

getline(ifs, raw\_string);

str\_start = raw\_string.find('(') + 1; // gets position of the start of the pic\_name

str\_end = raw\_string.find(',') - 1; // gets position of the end of the pic\_name

pic\_name = raw\_string.substr(str\_start, str\_end);

index\_read.push\_back(pic\_name);

}

ifs.close();

draw\_image(index\_read[index]);

}

}

}

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

void Display\_Window::quit()

{

hide();

}

void Display\_Window::home()

{

//Changes the browsing status, indicating that you can see all images

ostringstream ss;

ss<<"All Images";

mode.put(ss.str());

search\_mode = false;

ostringstream clean\_box;

clean\_box << " ";// clears the tags box

tags\_displayed.put(clean\_box.str()); // clears the tags displayed

family\_i = 0;

friends\_i = 0;

aggieland\_i = 0;

pets\_i = 0;

vacation\_i = 0;

}

void Display\_Window::Display\_active\_tags()

{

int current\_tags[] = { family\_i, friends\_i, aggieland\_i, pets\_i, vacation\_i };

string tag\_aray[] = { "family", "friends", "aggieland", "pets", "vacation" };

string ts\_arr[4]; // each string

string tags\_to\_show = " ";

ostringstream active\_t\_stream;

active\_t\_stream << tags\_to\_show; // clear box

tags\_displayed.put(active\_t\_stream.str());

for (int k = 0; k <= 4; k++)

{

if (current\_tags[k] == 1)

{

ts\_arr[k] = tag\_aray[k] + " ";

}

if (current\_tags[k] == 0)

{

ts\_arr[k] = " ";

}

}

tags\_to\_show = ts\_arr[0] + ts\_arr[1] + ts\_arr[2] + ts\_arr[3] + ts\_arr[4];

active\_t\_stream << tags\_to\_show;

tags\_displayed.put(active\_t\_stream.str());

for (int l = 0; l <= 4; l++)

{

current\_tags[l] = 0; // clear array

}

}

void Display\_Window::tag0()

{

ostringstream clean\_box;

clean\_box << " ";// clears the tags box

tags\_displayed.put(clean\_box.str()); // clears the tags displayed

if (family\_i == 1)

{

family\_i = 0;

}

if (family\_i == 0)

{

family\_i = 1;

}

Display\_active\_tags();

}

void Display\_Window::tag1()

{

ostringstream clean\_box;

clean\_box << " ";// clears the tags box

tags\_displayed.put(clean\_box.str()); // clears the tags displayed

if (friends\_i == 1)

{

friends\_i = 0;

}

if (friends\_i == 0)

{

friends\_i = 1;

}

Display\_active\_tags();

}

void Display\_Window::tag2()

{

ostringstream clean\_box;

clean\_box << " ";// clears the tags box

tags\_displayed.put(clean\_box.str()); // clears the tags displayed

if (aggieland\_i == 1)

{

aggieland\_i = 0;

}

if (aggieland\_i == 0)

{

aggieland\_i = 1;

}

Display\_active\_tags();

}

void Display\_Window::tag3()

{

ostringstream clean\_box;

clean\_box << " ";// clears the tags box

tags\_displayed.put(clean\_box.str()); // clears the tags displayed

if (pets\_i == 1)

{

pets\_i = 0;

}

if (pets\_i == 0)

{

pets\_i = 1;

}

Display\_active\_tags();

}

void Display\_Window::tag4()

{

ostringstream clean\_box;

clean\_box << " ";// clears the tags box

tags\_displayed.put(clean\_box.str()); // clears the tags displayed

if (vacation\_i == 1)

{

vacation\_i = 0;

}

if (vacation\_i == 0)

{

vacation\_i = 1;

}

Display\_active\_tags();

}

void Display\_Window::search()

{

//Changes the browsing status to say "Search Results"

ostringstream ss;

ss << "Search Results";

mode.put(ss.str());

Tag\_obj current\_tags = Tag\_obj(family\_i, friends\_i, aggieland\_i, pets\_i, vacation\_i);

vector<string> results = f\_search(db\_filename, current\_tags);

ostringstream no\_match; // Prints message if no matches are found

if ((results[0] == "") && (results.size() == 1))

{

no\_match << "No matches";

tags\_displayed.put(no\_match.str());

}

search\_mode = true;

ostringstream clean\_box;

clean\_box << " ";// clears the tags box

tags\_displayed.put(clean\_box.str()); // clears the tags displayed

family\_i = 0;

friends\_i = 0;

aggieland\_i = 0;

pets\_i = 0;

vacation\_i = 0;

}

void Display\_Window::draw\_image(string fname)

{

detach(\*p);

p = new Image(Point(50, 50), fname);

attach(\*p);

//virtual FL\_Image \*copy(int W, int H);

redraw();

}

void Display\_Window::add\_file()

{

ofstream ofs;

ostringstream xx;

string URLstring = input\_url.get\_string();

string file\_name = input\_file.get\_string();

if(family\_i==1)

{

family\_s = "family";

}

else

{

family\_s = " ";

}

if(friends\_i==1)

{

friends\_s = "friends";

}

else

{

friends\_s = " ";

}

if(aggieland\_i==1)

{

aggieland\_s = "aggieland";

}

else

{

aggieland\_s = " ";

}

if(pets\_i==1)

{

pets\_s = "pets";

}

else

{

pets\_s = " ";

}

if(vacation\_i==1)

{

vacation\_s = "vacation";

}

else

{

vacation\_s = " ";

}

if ((int)URLstring.find("http")>-1)//if a URL exists

{

xx<<" ";// clears the tags box

tags\_displayed.put(xx.str()); // clears the tags displayed

system((string("wget -O " + file\_name + " " + URLstring).c\_str()));

family\_i = 0;

friends\_i = 0;

aggieland\_i = 0;

pets\_i = 0;

vacation\_i = 0;

}

// you can use to\_lower so that you don't have to put different cases here

if (file\_name.substr(file\_name.find\_last\_of(".") + 1) == "jpg" || file\_name.substr(file\_name.find\_last\_of(".") + 1) == "jpeg" || file\_name.substr(file\_name.find\_last\_of(".") + 1) == "gif" || file\_name.substr(file\_name.find\_last\_of(".") + 1) == "JPG" || file\_name.substr(file\_name.find\_last\_of(".") + 1) == "JPEG" || file\_name.substr(file\_name.find\_last\_of(".") + 1) == "GIF" )

{

xx<<" ";// clears the tags box

ofs.open(db\_filename, fstream::app);

ofs << "(" << file\_name << ',' << family\_s << ',' << friends\_s << ',' << aggieland\_s << ',' << pets\_s << ',' << vacation\_s << "\n";

ofs.close();

family\_i = 0;

friends\_i = 0;

aggieland\_i = 0;

pets\_i = 0;

vacation\_i = 0;

draw\_image(file\_name);

} // you can use to\_lower so that you don't have to put different cases here

else //wrong file type

{

//Error\_window(Point(0,0), 500, 200, "Error!");

}

}

int main()

{

try

{

if (H112 != 201401L)error("Error: incorrect std\_lib\_facilities\_4.h version ", H112);

Display\_Window w(Point(100,100),800,600,"Aggie Snap!");

Intro\_Window i(Point(100,100),800,600,"Instructions");

return gui\_main();

}

catch (exception& e)

{

cerr << "exception: " << e.what() << '\n';

return 1;

}

catch (...)

{

cerr << "Some exception\n";

return 2;

//testing

}

}

**/////////////////////////////////////////////////////////////////////////**

**/Input.cpp////////////////////////////////////////////////////////////**

**/////////////////////////////////////////////////////////////////////////**

#include "universal.h"

vector<string> Display\_Window::f\_search(string db\_fname, Tag\_obj t)

{

ifstream ist(db\_fname.c\_str());

string raw\_string, pic\_name;

int str\_start, str\_end, found;

bool contains\_tag; // if any of the tags found, push to the results

char trash\_newlines;

bool btag\_aray[] = { t.family, t.friends, t.aggieland, t.pets, t.vacation };

string tag\_aray[] = { "family", "friends", "aggieland", "pets", "vacation" };

vector<string> s\_results;

if (!ist) error("can't open input file", db\_fname);

// Read entire line,parse each line, stop if end of file is found

while (true)

{

if (ist.eof()) break; // exit if end of file

getline(ist, raw\_string);

for (int i = 0; i <= 4; i++)

{

if (btag\_aray[i]){ found = raw\_string.find(tag\_aray[i]); }

else { found = -1; }

if (found >= 0) //any tag is present

{

contains\_tag = true;

str\_start = raw\_string.find('(') + 1;

str\_end = raw\_string.find(',') - 1;

pic\_name = raw\_string.substr(str\_start, str\_end);

}

}

if (contains\_tag)

{

s\_results.push\_back(pic\_name);

contains\_tag = false;

found = -1;

}

}

if ((s\_results[0] == "") && (s\_results.size() == 1)){ cout << "No matches.\n"; }

return s\_results;

}

**/////////////////////////////////////////////////////////////////////////**

**/Input.cpp////////////////////////////////////////////////////////////**

**/////////////////////////////////////////////////////////////////////////**

#include "universal.h"

Error\_window::Error\_window(Point xy, int w, int h, const string& title)

:Window(xy,w,h,title),

message(Point(0,100), "Incorrect input! Try again."),

ok\_button(Point(x\_max()-100,0),100,25, "OK", cb\_ok)

{

attach(message);

attach(ok\_button);

}

void Error\_window::cb\_ok(Address, Address pw)

{

reference\_to<Error\_window>(pw).ok();

}

void Error\_window::ok()

{

hide();

}