**Flip Nation: A Pancake Game**

**Group 26: CSCE 121-501**

Andrew Garcia-Senior Developer,

Christopher Comeaux-Project Leader and Submitter

Tyler Creamer-Debugger and Coordinator

Throughout the project, some tasks required more time to write and debug, and some required more time to research. As far as we could tell, all contributions were roughly equal.

In this project we were tasked with creating a game. The nature of this game is the famous pancake sorting problem. The player chooses a number of pancakes, from 2 to 9, and the game generates a stack of pancakes. The player must then sort the pancakes from smallest to largest by flipping different sections the stack. The player is given a score depending on the number of flips it took them to complete the puzzle and the difficulty chosen. They are then asked to input their initials and the 5 highest scores are displayed. If they flip too many pancakes and their score becomes negative, they are given a notification and asked if they want to try again. This problem is significant in that it requires a great diversity of operations to complete. To complete the program, a graphics window, display text, a vector of random numbers such that no 2 are the same, and much more all had to be written in code and implemented correctly.

Our biggest restriction was time. We would have liked to make the project more visually exciting, but because of time, we could not. Our second restriction was the detailed project parameters. If we had more freedom with how the pancakes looked, we would have used overlapping circles and rectangles to create a more real life looking pancake. We planned to use a tan fill color and an invisible line color. The biggest limitation was, of course, our knowledge. We spent more time trying to figuring out how to write the code than actually writing and debugging it.

In order to create a project, planning is critical. As such, we started out with an outline and met periodically to go over what we had done and plan out future steps. We analyzed the assignment, and broke it down into a list of smaller tasks that we distributed among ourselves. The division of tasks was not based on difficulty, but focused on different functionalities of the project. For example, we delegated buttons to one member and score calculation to another.

Sample run (screen shots)

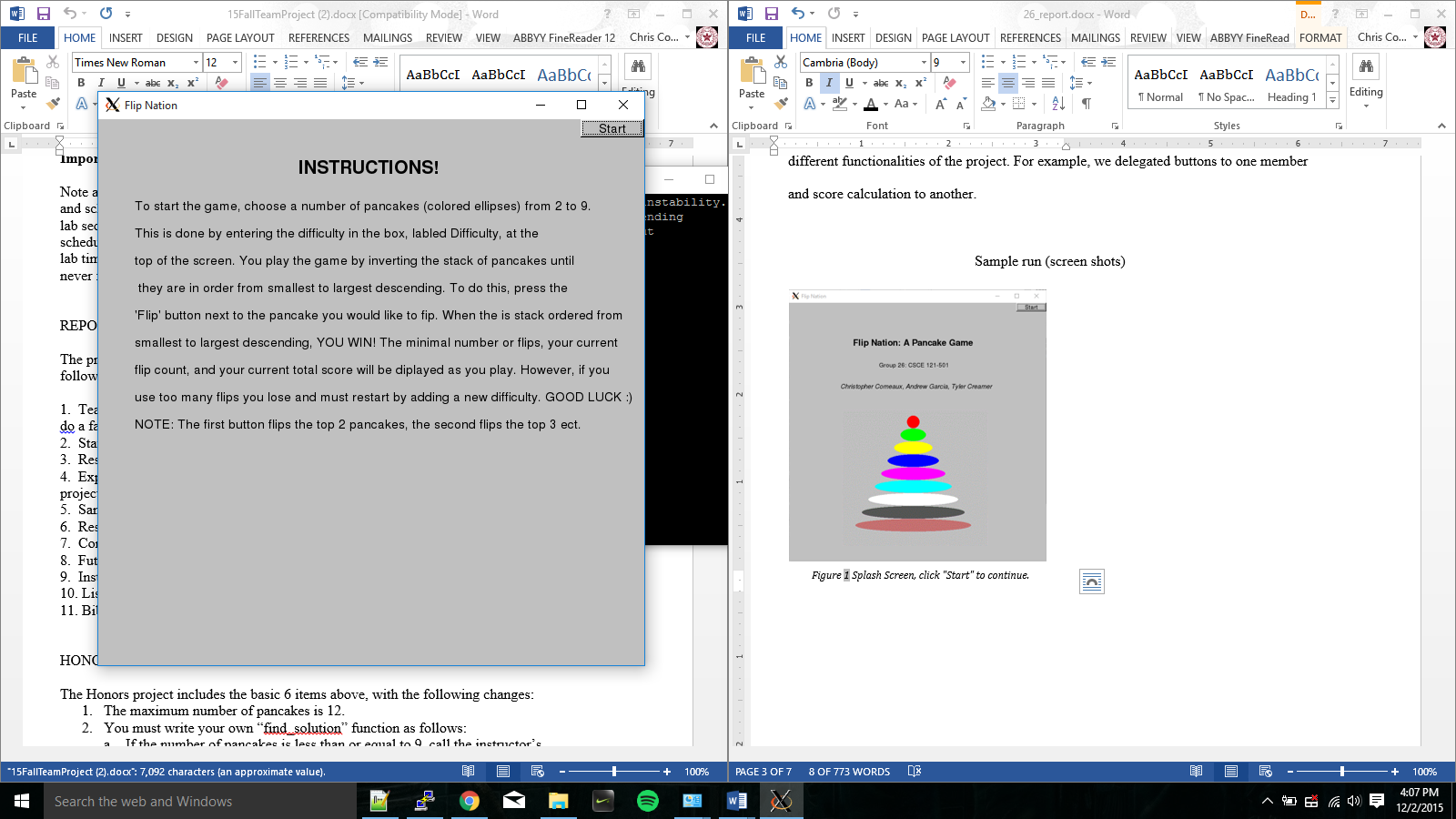
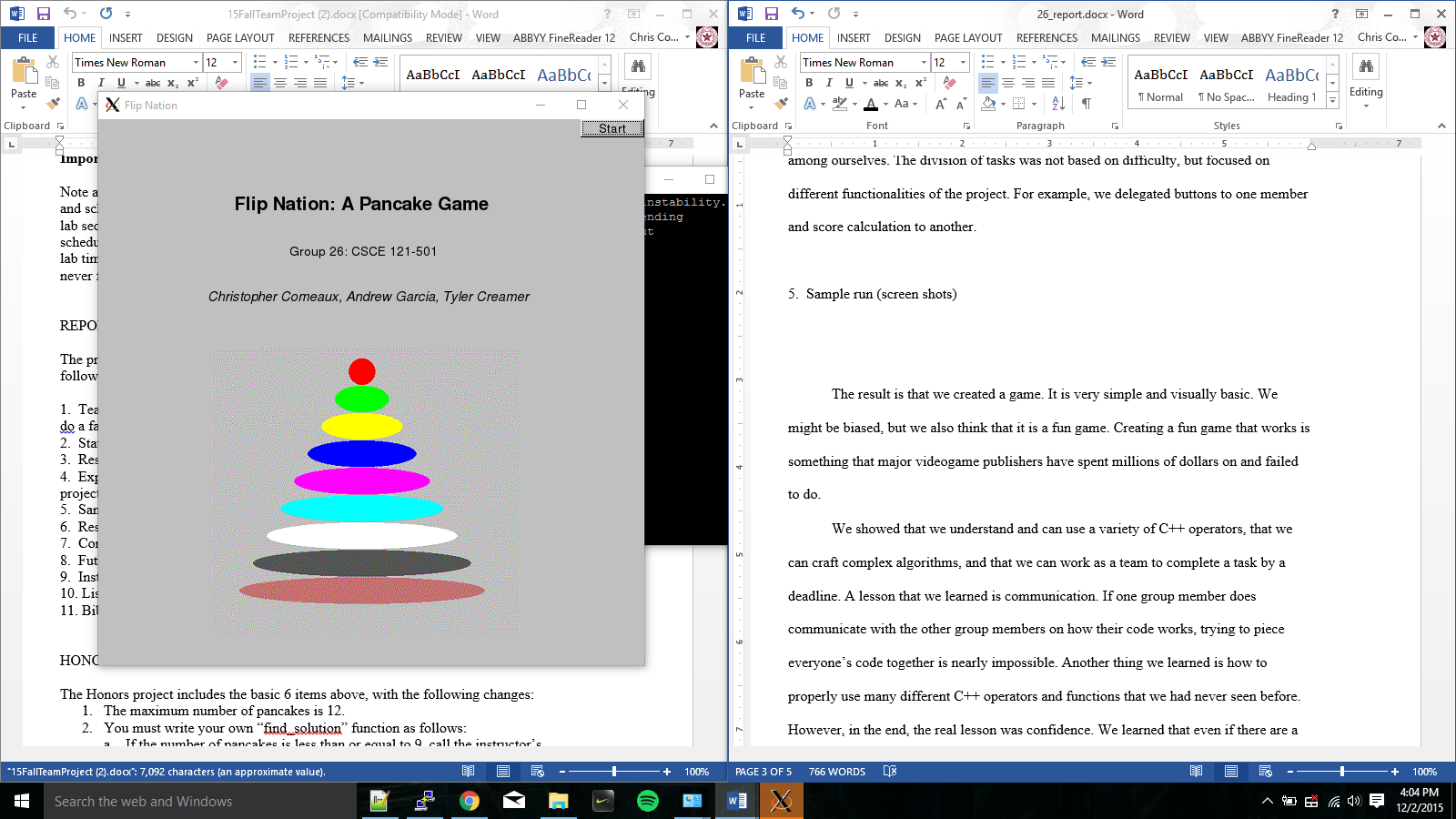


Figure 2 Instructions Screen, Click "Start" to continue

Figure 1 Splash Screen, click "Start" to continue.

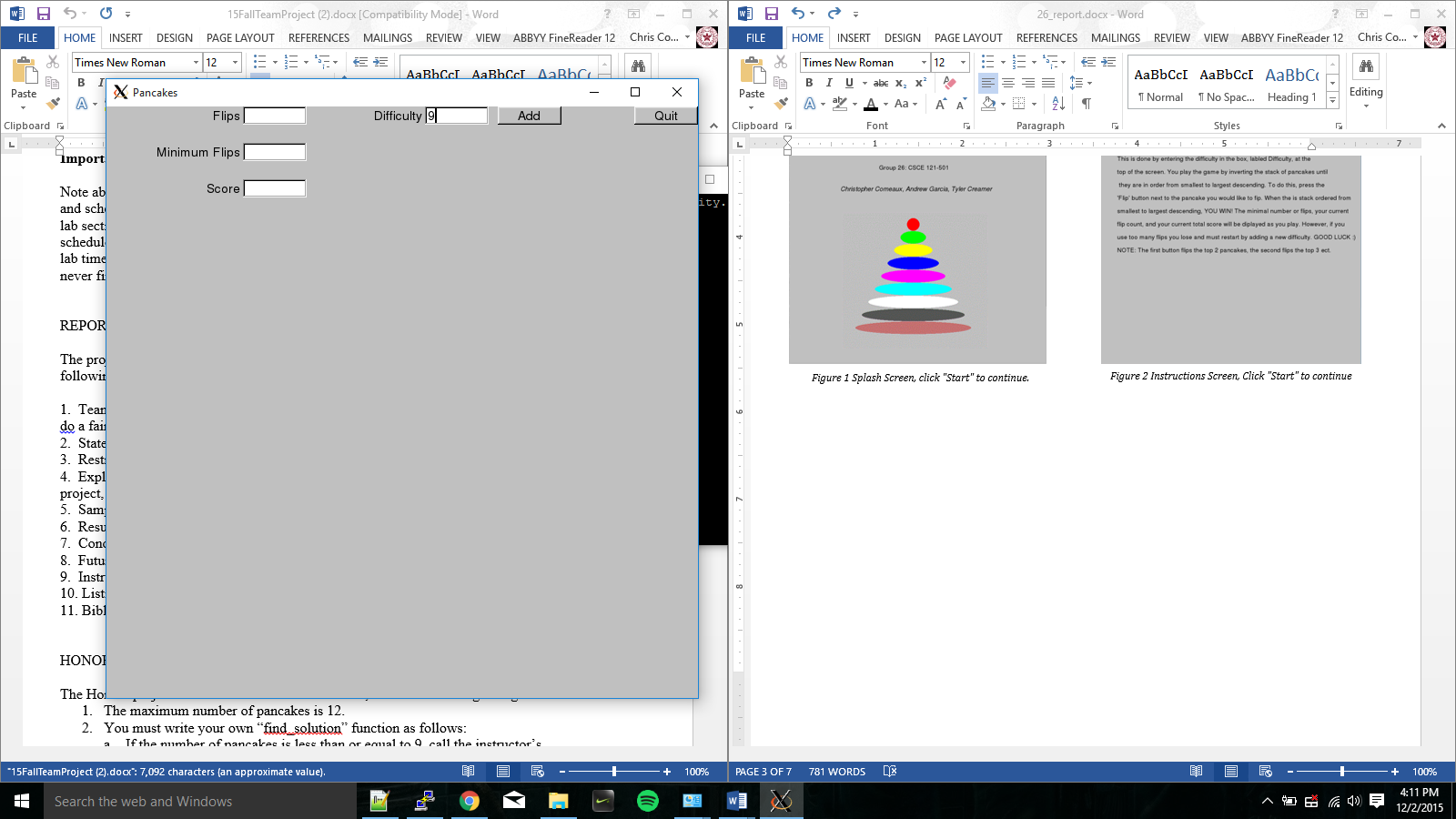


Figure 3 Game Screen, enter a difficulty and press "Add“ to start the game

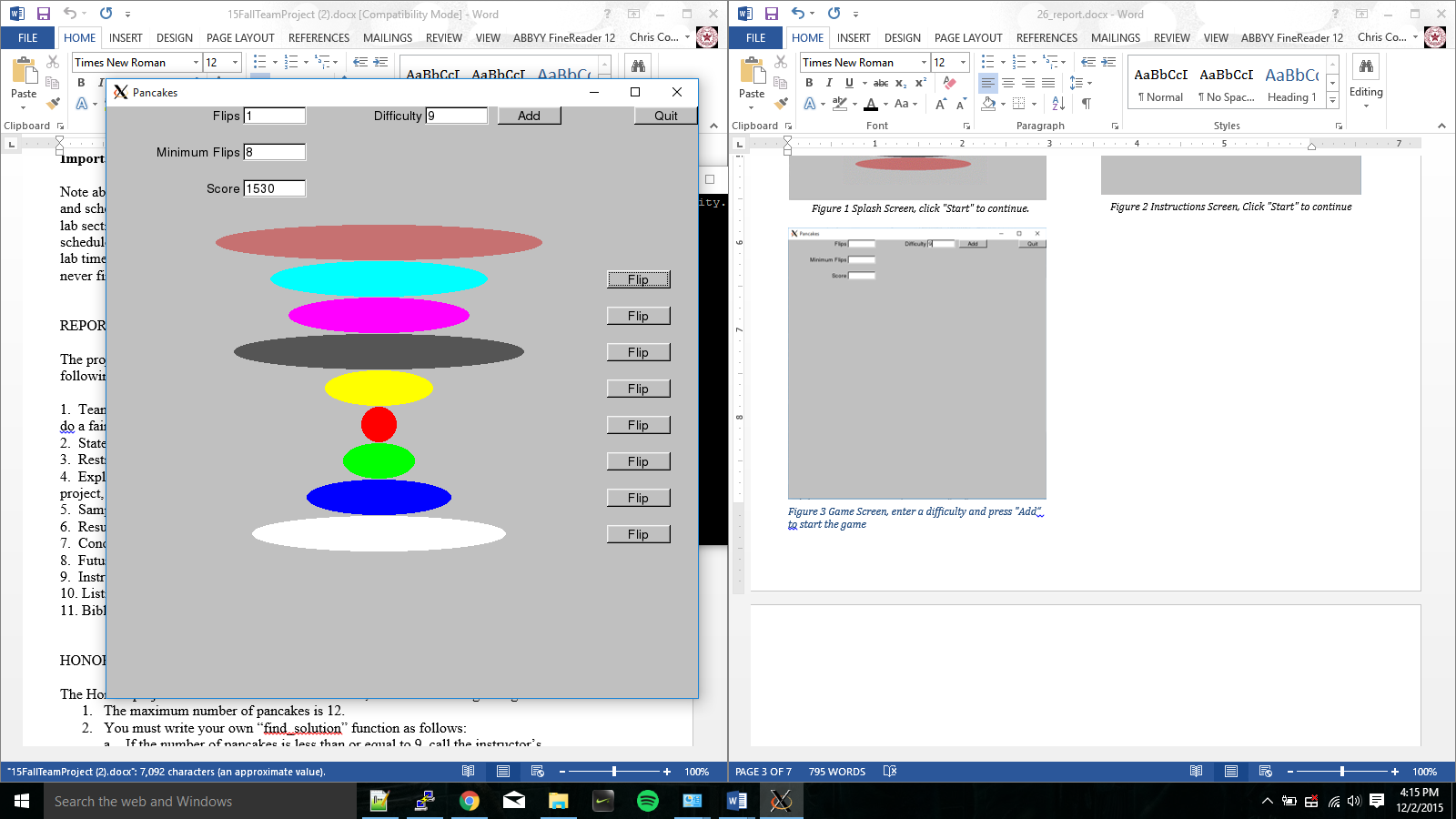


Figure 4 In Game, Use "Flip" buttons to rearrange pancakes. The data described in the instructions is displayed in the top left

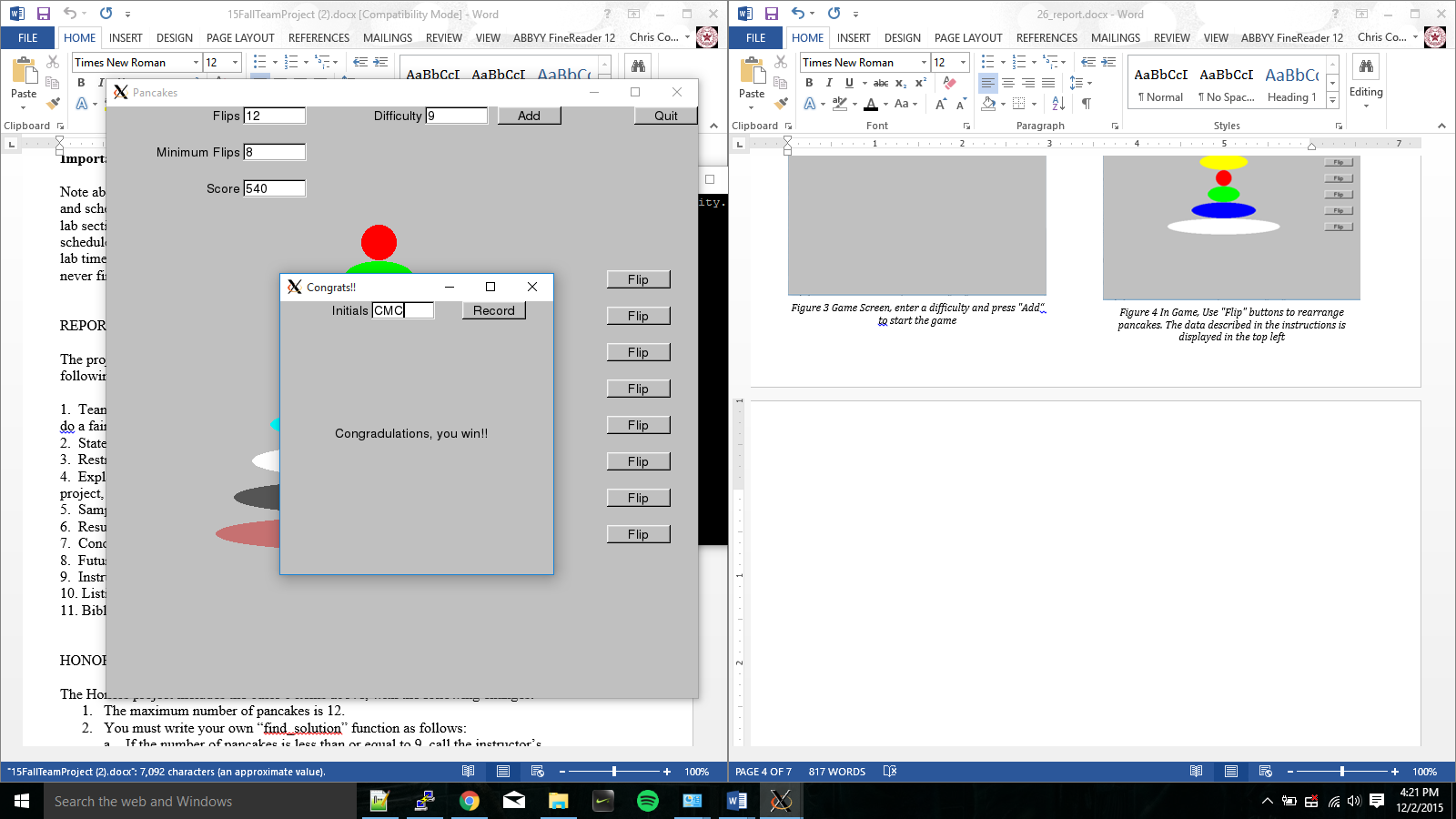
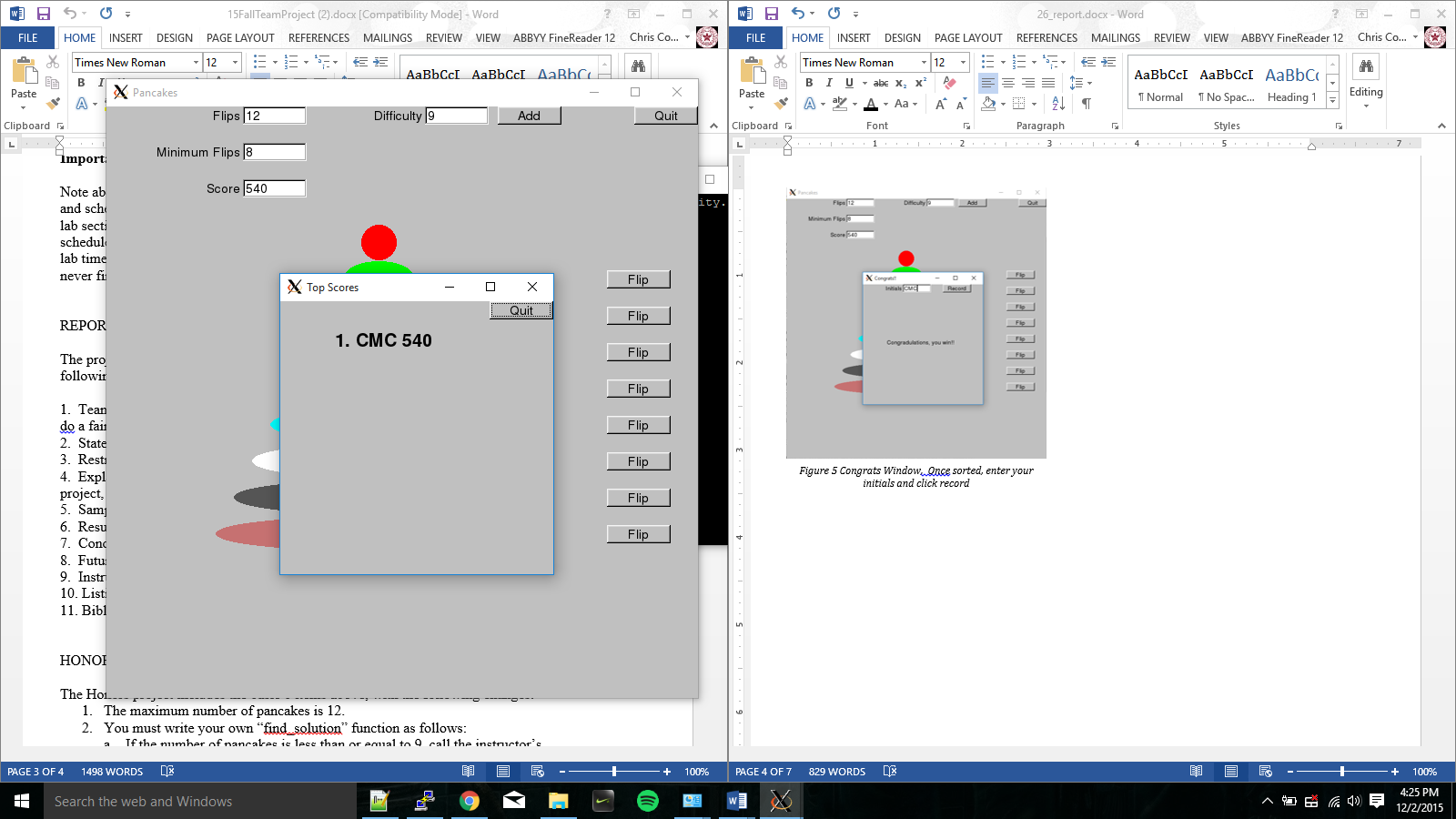


Figure 6 Top Scores, click the "quit" button to close the scores window. If you want to play again refer to Figure 3. Otherwise click "Quit" in the "Pancakes" window to close the program

Figure 5 Congrats Window, Once sorted, enter your initials and click record

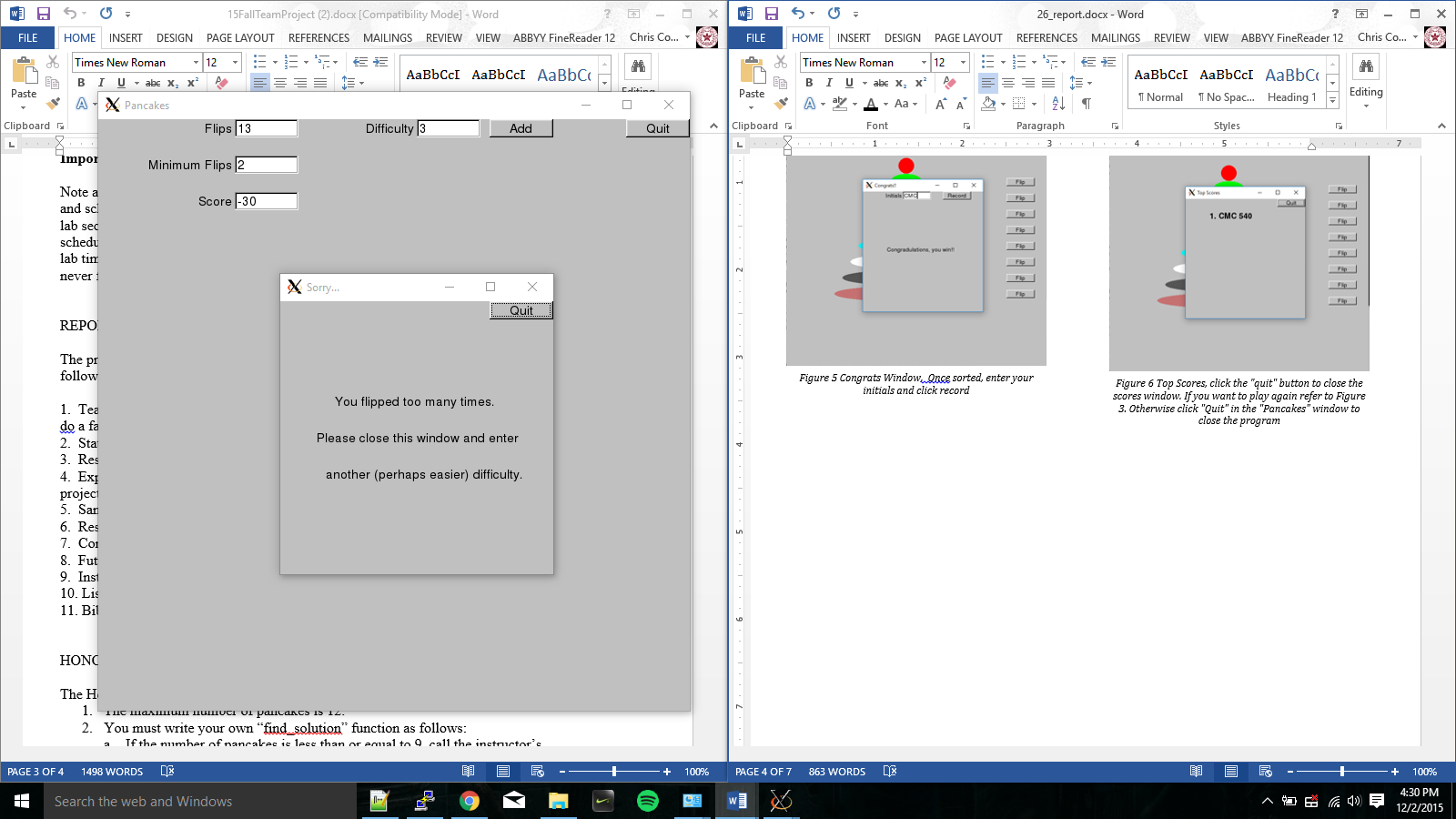


Figure 7 Error Window, If you lose, click "Quit" on the “sorry…” window. Then you can either choose to restart the game (refer to Figure 3) or close the game by clicking "Quit" in the" Pancakes" window

The result is that we created a game. It is very simple and visually basic. We might be biased, but we also think that it is a fun game. Creating a fun game that works is something that major videogame publishers have spent millions of dollars on and failed to do.

We showed that we understand and can use a variety of C++ operators, that we can craft complex algorithms, and that we can work as a team to complete a task by a deadline. A lesson that we learned is communication. If one group member does communicate with the other group members on how their code works, trying to piece everyone’s code together is nearly impossible. Another thing we learned is how to properly use many different C++ operators and functions that we had never seen before. However, in the end, the real lesson was confidence. We learned that even if there are a lot of unknowns, a group of diligent people can figure it out together.

There are many ways that our program could be improved. For example, the visuals could be more evocative. We could have added animations showing how to play the game, or simply have added more color in the background. Another improvement would have been to allow the user to select the buttons by using the arrow keys and the “Enter” button on the keyboard.

In order to run the program place the CD in the disk drive of a computer. Save the folder to on the CSCE server. Open PuTTY and navigate to the SSH tab. Click X11 and select the box that says “Enable X11 forwarding”. Also, you must have Xming open. Then log into the server and navigate to the folder using the terminal. Next enter the command “g++ -std=c++14 \*cpp \*o –lfltk –lfltk\_images” to compile the project. Once this is done, use “.\a.out” to run the program.

Listing of the COMMENTED program

**PWINDOW.CPP**

#include "Misc\_Windows.h"

int main()

{

Splash\_Window win00(Point(100,100),600,600,"Flip Nation");

return gui\_main();

}

PANCAKE.H

#include "Graph.h"

#include "GUI.h"

using namespace Graph\_lib;

struct Pancake : Shape

{

Pancake(int width, int position); //constructor

void draw\_lines() const;

void move(int dx, int dy);

int len; int pos;

Ellipse pan{Point{300,pos\*40+150},len\*20, 20};

};

**PANCAKE.CPP**

#include"Pancake.h"

/\*------------------------------------------------------------------------------------------

PANCAKE IMPLEMENTAION\*/

Pancake::Pancake(int width, int position)

:len{width}, pos{position}

{

pan.set\_fill\_color(len); //setting colors and fills for each of the pancakes

pan.set\_color(Color::invisible);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void Pancake::draw\_lines() const

{

pan.draw();

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void Pancake::move(int dx, int dy)

{

pan.move(dx, dy);

}

**PANCAKE\_WIN.H**

#include "Window.h"

#include "Graph.h"

#include "GUI.h"

#include "Pancake.h"

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

//PancakeWin.h

//Header file for Pancake window, Congrats Window, and Scores Window

struct Pancake\_Window : Graph\_lib::Window

{

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

private:

Button next\_button;

Button quit\_button;

In\_box user\_input;

Out\_box flip\_output;

Out\_box score\_output;

Out\_box min\_flips;

Vector\_ref<Pancake> vp; //vector of pancakes

Vector\_ref<Button> vb; //vector of buttons

vector<int> pan\_orig; //the order of the pancakes

vector<int> sorted; //holds the sorted vector of ints to test against

int num\_pan; //holds the user input recieved through the user\_inout In\_Box

int min; //holds the minimum number of flips the puzzle can be completed in

int score; //calculates the score

//actions invoked by callbacks

void quit();

int next();

int flip\_pressed();

//helper functions

void add\_buttons();

void add\_pancakes();

void detach\_buttons();

void detach\_pancakes();

int out\_of\_range();

int is\_score\_neg();

int get\_button\_num();

//callback functions

static void cb\_flip(Address, Address pw) {reference\_to<Pancake\_Window>(pw).flip\_pressed();};

static void cb\_next(Address, Address pw) {reference\_to<Pancake\_Window>(pw).next();};

static void cb\_quit(Address, Address pw) {reference\_to<Pancake\_Window>(pw).quit();};

};

struct Congrats\_Window : Graph\_lib::Window

{

Congrats\_Window(Point xy, int w, int h, const string& title); //constructor

private:

Text congrats; //congradulations text that get displayed on the screen

In\_box initials; // gets users initials

string initial; // hold the users initials

Button Record; // button to record the initials

//used to calculate scores and output them

ifstream input\_scores; //linked to Scores.txt

ofstream output\_scores;

vector<string> init\_vect;

vector<int> scores\_vect;

//functions used to handle scoring and recorinf scores

void scores\_input();

void sort\_input();

void scores\_output();

//functions and callbacks

int record();

static void cb\_record (Address, Address pw) {reference\_to<Congrats\_Window>(pw).record();};

};

struct Scores\_Window : Graph\_lib::Window

{

Scores\_Window(Point xy, int w, int h, const string& title); // constructor

private:

Text\* scores; // used to display scores on the window

Button Quit\_Button;

ifstream scores\_stream; // linked to Scores.txt

vector<string> initial\_vector;

vector<int> scores\_vector;

//helper functions

void fill\_vectors(); // fill the vectors listed above and display Text scores

// Functions and callbacks

void quit\_scores();

static void cb\_quit\_scores (Address, Address pw) {reference\_to<Scores\_Window>(pw).quit\_scores();};

};

struct Error\_Window : Graph\_lib::Window

{

Error\_Window(Point xy, int w, int h, const string& title); // constructor

private:

Button Quit\_Button;

//Functions and their callbacks

void quit\_error();

static void cb\_quit\_error (Address, Address pw) {reference\_to<Error\_Window>(pw).quit\_error();};

};

**PANCAKE\_WIN.CPP**

#include "PancakeWin.h"

#include "find\_solution.h"

/\* GLOBAL VERIABLES\*/

int total\_score; // global variable to access private members

string current\_initials; // global variable to access provate members

/\*---------------------------------------------------------------------

PANCAKE\_WINDOW IMPLEMENTATION\*/

//Constructor

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

:Window{xy,w,h,title},

next\_button{Point{x\_max()-220,0}, 70, 20, "Add", cb\_next},

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

user\_input{Point{x\_max()-300,0}, 70, 20, "Difficulty"},

flip\_output{Point{x\_max()-500,0}, 70, 20, "Flips"},

score\_output{Point{x\_max()-500,80}, 70, 20, "Score"},

min\_flips{Point{x\_max()-500,40}, 70, 20, "Minimum Flips"}

{

attach(next\_button);

attach(quit\_button);

attach(user\_input);

attach(flip\_output);

attach(score\_output);

attach(min\_flips);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//hide PANCAKE\_WINDOW

void Pancake\_Window::quit()

{

hide();

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//shuffle and display pancakes and buttons (ADD BUTTON)

int Pancake\_Window::next()

{

detach\_pancakes();

detach\_buttons();

num\_pan = user\_input.get\_int(); //set the number of pancakes to the users input

out\_of\_range();// if the user inputs a a number out of range display error window

pan\_orig.resize(num\_pan); // resize to the user input

for (int i = 0; i < pan\_orig.size(); ++i) //set each number to it's position {1,2,3...}

pan\_orig[i] = i + 1;

sorted = pan\_orig;// create a copy of pan\_orig before it is scrambled. To be used to test if player has won

srand(time(0)); //shuffle order vector

random\_shuffle(pan\_orig.begin(), pan\_orig.end());

//find minimal solution

vector<int> solution = \*find\_solution(pan\_orig);

min = solution.size();

min\_flips.put(to\_string(min));//display the minimal number of flips

add\_pancakes();

add\_buttons();

if( min == 0) //if the pancakes are already sorted deconstruct vectors and rescrable

next();

score=0; //reset the score user adds a new set of pancakes

redraw();

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//flip pancakes -- The pancake are flipped based on position

// so the first button flips the first two and so on

int Pancake\_Window::flip\_pressed()

{

vector<int> pan\_flip = pan\_orig; //make another vector identical to the order vector

int last = get\_button\_num(); //this determines how many pancakes will be flipped

int first = 0;

int flipped = last;

while (first < last) // flips the pancakes in the vector

{

pan\_flip[first] = pan\_orig[flipped - 1];

++first;

--flipped;

}

pan\_orig = pan\_flip;

detach\_pancakes();

add\_pancakes();

//keep the user updated on how they are doing

score++; //updates score

flip\_output.put(to\_string(score));

total\_score = (100-10\*(score-min))\* num\_pan;

score\_output.put(to\_string(total\_score));

is\_score\_neg();// if they use too many flips open error window

redraw();

if(pan\_orig==sorted) // if the user wins display the Congrats window and record initials this

{

Congrats\_Window win02{Point{300,300},300,300,"Congrats!!"};

return gui\_main();

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void Pancake\_Window::add\_buttons()

{

for (int i = 0; i < pan\_orig.size()-1; ++i) //create vector of buttons and attach them

{

vb.push\_back(new Button{Point{x\_max()-100,i\*40+180}, 70, 20, "Flip", cb\_flip});

attach(vb[vb.size()-1]);

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void Pancake\_Window::add\_pancakes()

{

for (int i = 0; i < user\_input.get\_int(); ++i) //place a pancake of size pan\_orig[i] at position i

{

vp.push\_back(new Pancake{pan\_orig[i], i});

attach(vp[vp.size()-1]);

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void Pancake\_Window::detach\_buttons()

{

for (int i = 0; i < vb.size(); ++i) //detach any previous buttons

detach(vb[i]);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void Pancake\_Window::detach\_pancakes()

{

for (int i = 0; i < vp.size(); ++i) //detach any previous pancakes

detach(vp[i]);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

int Pancake\_Window::out\_of\_range()

{

if (num\_pan < 2 || num\_pan > 9)

{

Error\_Window win03{Point{300,300},300,300,"Sorry..."};

Text dummy1{Point{45,115}, "Did you even look at the instructions?"};

Text dummy2{Point{45,155}, "Please close this window and enter a"};

Text dummy3{Point{70,195}, "difficulty between 2 and 9."};

win03.attach(dummy1);

win03.attach(dummy2);

win03.attach(dummy3);

return gui\_main();

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//test to see if player used too many flips

int Pancake\_Window::is\_score\_neg()

{

if (total\_score < 0)

{

Error\_Window win03{Point{300,300},300,300,"Sorry..."};

Text loser1{Point{60,115}, "You flipped too many times."};

Text loser2{Point{40,155}, "Please close this window and enter"};

Text loser3{Point{50,195}, "another (perhaps easier) difficulty."};

win03.attach(loser1);

win03.attach(loser2);

win03.attach(loser3);

detach\_buttons();

detach\_pancakes();

return gui\_main();

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// the position of the button deterimes how many pancakes will be flipped

int Pancake\_Window::get\_button\_num()

{

int y = Fl::event\_y(); // find where the user clicked, used to control the number of panckakes flipped

int n; //represents how many pancakes will be flipped--Ex. n=9 means all 9 pancakes will be flipped

if (y >=460 && y <=480) n=9;

else if (y >=420 && y <=460) n=8;

else if (y >=380 && y <=400) n=7;

else if (y >=340 && y <=360) n=6;

else if (y >=300 && y <=320) n=5;

else if (y >=260 && y <=280) n=4;

else if (y >=220 && y <=240) n=3;

else n=2;

return n;

}

/\*-----------------------------------------------------------------------

CONGRATS WINDOW IMPLEMENTAION\*/

//Constructor

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

:Window{xy,w,h,title},

congrats{Point{60,150}, "Congradulations, you win!!"},

initials{Point{x\_max()-200,0}, 70, 20, "Initials"},

Record{Point{x\_max()-100,0}, 70, 20, "Record", cb\_record}

{

attach(congrats);

attach(initials);

attach(Record);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//record the users initials and display the Scores Window

int Congrats\_Window::record()

{

hide();

initial = initials.get\_string();

current\_initials = initials.get\_string();

if(initial.length() > 3)// if initials are too long conctinate it to size 3

initial.resize(3);

scores\_output();

Scores\_Window win20{Point{300,300},300,300,"Top Scores"};

return gui\_main();

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//input scores from the file names Scores.txt

void Congrats\_Window::scores\_input()

{

input\_scores.open("Scores.txt");

if (!input\_scores)

error("cannot open file Scores.txt");

string I; // holds the initials

int S; //holds the scores

while(input\_scores >> I >> S)

{

init\_vect.push\_back(I);

scores\_vect.push\_back(S);

}

input\_scores.close();

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//sort into highest score

void Congrats\_Window::sort\_input()

{

scores\_input(); // get input from file

// obtain most recent scores

init\_vect.push\_back(initial);

scores\_vect.push\_back(total\_score);

for (int c = 0; c < scores\_vect.size()-1; c++)

{

for(int i = c+1; i<scores\_vect.size(); i++)

{

if(scores\_vect[c] < scores\_vect[i])

{

int tmpi = scores\_vect[i];

string tmps = init\_vect[i];

scores\_vect[i] = scores\_vect[c];

init\_vect[i] = init\_vect[c];

init\_vect[c] = tmps;

scores\_vect[c] = tmpi;

}

}

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// sort and output scores back to Scores.txt

void Congrats\_Window::scores\_output()

{

sort\_input();

output\_scores.open("Scores.txt");

if(!output\_scores)

error("cannot open file Scores.txt");

int size = 5;

if(init\_vect.size() < 5)

size = init\_vect.size();

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

{

output\_scores << init\_vect[i] + " " + to\_string(scores\_vect[i]) + '\n';

}

output\_scores.close();

}

/\*-------------------------------------------------------------------------------

SCORES WINDOW IMPLEMETAION\*/

//Constructor

Scores\_Window::Scores\_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\_scores}

{

fill\_vectors();

attach(Quit\_Button);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//close the Scores Window

void Scores\_Window::quit\_scores()

{

hide();

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Fill the vectors that hold the data inputed from Scores.txt

//and represent it on the scren as text objects

void Scores\_Window::fill\_vectors()

{

//fill the vectors

scores\_stream.open("Scores.txt");

if (!scores\_stream)

error("cannot open file Scores.txt");

string I; // holds the initials

int S; //holds the scores

while(scores\_stream >> I >> S)

{

initial\_vector.push\_back(I);

scores\_vector.push\_back(S);

}

//display the data as text

int y = 50;

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

{

scores = new Text{Point{60,y}, to\_string(i+1) + ". " + initial\_vector[i] + "\t" + to\_string(scores\_vector[i])};

attach(\*scores);

scores->set\_font\_size(20);

if( initial\_vector[i] == current\_initials && scores\_vector[i] == total\_score) // sets current players score to bold if they are one of the top scores

scores->set\_font(FL\_HELVETICA\_BOLD);

y += 50;

}

scores\_stream.close();

}

/\*-----------------------------------------------------------------------------------------------------------

ERROR WINDOW IMPLEMENTAION\*/

Error\_Window::Error\_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\_error}

{

attach(Quit\_Button);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void Error\_Window::quit\_error()

{

hide();

}

**MISC\_WINDOWS.H**

#include "PancakeWin.h"

struct Splash\_Window : Graph\_lib::Window

{

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

private:

Text names;

Text section;

Text Title;

Image Stack;

Button start\_button;

int start(); // go to the intructional screen

static void cb\_start (Address, Address pw) {reference\_to<Splash\_Window>(pw).start();};

};

struct Instruction\_Window : Graph\_lib::Window

{

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

private:

Text instructions;

Text explnation0;

Text explnation1;

Text explnation2;

Text explnation3;

Text explnation4;

Text explnation5;

Text explnation6;

Text explnation7;

Text explnation8;

Button Start\_button; // start the game

int Start();

static void cb\_Start (Address, Address pw) {reference\_to<Instruction\_Window>(pw).Start();};

};

**MISC\_WINDOWS.CPP**

#include "Misc\_Windows.h"

/\*----------------------------------------------------------------------------------

SPLASH WINDOW IMPLEMENTATION\*/

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

:Window{xy,w,h,title},

names {Point{120,200},"Christopher Comeaux, Andrew Garcia, Tyler Creamer"},

section {Point{210,150},"Gorup 26: CSCE 121-501"},

Title {Point{150,100},"Flip Nation: A Pancake Game"},

start\_button {Point{530,0},70, 20, "Start",cb\_start},

Stack{Point{125,250},"stack.gif"}

{

Title.set\_font\_size(20);

Title.set\_font(FL\_BOLD);

names.set\_font\_size(15);

names.set\_font(FL\_ITALIC);

attach(names);

attach(section);

attach(Title);

attach(start\_button);

attach(Stack);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

int Splash\_Window::start() //closes the splash window and brings up the instructions

{

hide();

Instruction\_Window win01(Point(100,100),600,600,"Flip Nation");

return gui\_main();

}

/\*---------------------------------------------------------------------------------------------------------

INSTRUCTION WINDOW IMPLEMENTATION\*/

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

:Window{xy,w,h,title},

instructions {Point{220,60}, "INSTRUCTIONS!"},

explnation0 {Point{40,100},"To start the game, choose a number of pancakes (colored ellipses) from 2 to 9."},

explnation1 {Point{40,130},"This is done by entering the difficulty in the box, labeled Difficulty, at the "},

explnation2 {Point{40,160},"top of the screen. You play the game by inverting the stack of pancakes until"},

explnation3 {Point{40,190}," they are in order from smallest to largest descending. To do this, press the "},

explnation4 {Point{40,220},"'Flip' button next to the pancake you would like to flip. When the is stack ordered from"},

explnation5 {Point{40,250},"smallest to largest descending, YOU WIN! The minimal number or flips, your current"},

explnation6 {Point{40,280},"flip count, and your current total score will be displayed as you play. However, if you"},

explnation7 {Point{40,310},"use too many flips you lose and must restart by adding a new difficulty. GOOD LUCK :)"},

explnation8 {Point{40,340},"NOTE: The first button flips the top 2 pancakes, the second flips the top 3 ect."},

Start\_button{Point{530,0},70, 20, "Start",cb\_Start}

{

instructions.set\_font\_size(20);

instructions.set\_font(FL\_HELVETICA\_BOLD);

attach(instructions);

attach(explnation0);

attach(explnation1);

attach(explnation2);

attach(explnation3);

attach(explnation4);

attach(explnation5);

attach(explnation6);

attach(explnation7);

attach(explnation8);

attach(Start\_button);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

int Instruction\_Window::Start()

{

hide();

Pancake\_Window win{Point{100,100},650,650,"Pancakes"}; // closes the instructions and brings up the game window

return gui\_main();

}

Bibliography

1. Stroustrup, Bjarne. Programming: Principles and Practice Using C. N.p.: n.p., n.d. Print.

2. ""C and C++ Reference." Cppreference.com. N.p., n.d. Web. 01 Dec. 2015. <http://en.cppreference.com/w/>.

3. "6 - Handling Events." FLTK 1.0.10 Programming Manual. N.p., n.d. Web. 01 Dec. 2015. <http://www.fltk.org/doc-1.0/events.html>.