REPORT OUTLINE

PART I – Team Information

The members of Team 4 (‘Please Compile’) are: Luke Grammer, Seungjin Kim, and Daniel Hain-Trevino.

Luke Grammer implemented the Window\_manager, Main\_window, Hint, and Help\_window classes, as well as Main.cpp.

Seungjin Kim implemented the Game, Game\_window, Tile, and User\_score classes.

Daniel Hain-Trevino implemented the Difficulty\_window, Initials\_window, and Game\_over\_window classes, and also one extra item (multiple randomly selected tilesets for each difficulty level)

Each team member did a fair share of work for this project.

PART II – The Problem

The problem was to create a functional 15 puzzle game using the FLTK graphics library in C++, being sure to implement a main menu, four different difficulty settings, a top 5 scores system using user-entered initials, and a hint button. The difficulty should determine how many moves the player receives, as well as the initial board layout. The score should be calculated as the number of moves taken multiplied by the tiles in their correct position. The hint functionality should use the Manhattan distance formula for each of the possible moves, and tell the player the move resulting in the smallest distance. The moves remaining as well as the number of incorrect tiles should be displayed to the screen during the game, and update accordingly. The top 5 scores should be displayed during the game, as well as written to a file for retrieval in subsequent games. A game over menu should display after all moves have been made, asking if the user wants to play another game or quit.

PART III – Restrictions and Limitations

The program needed to be written in C++, and implement FLTK graphics. Each function must fit inside of a standard console window (24 lines). The program must use at least two different features of C++11, C++14, or C++17. Each team member must write at least two classes. All input and output for the program must be through a GUI. The scores must be saved to a file and kept for use in subsequent runs of the program.

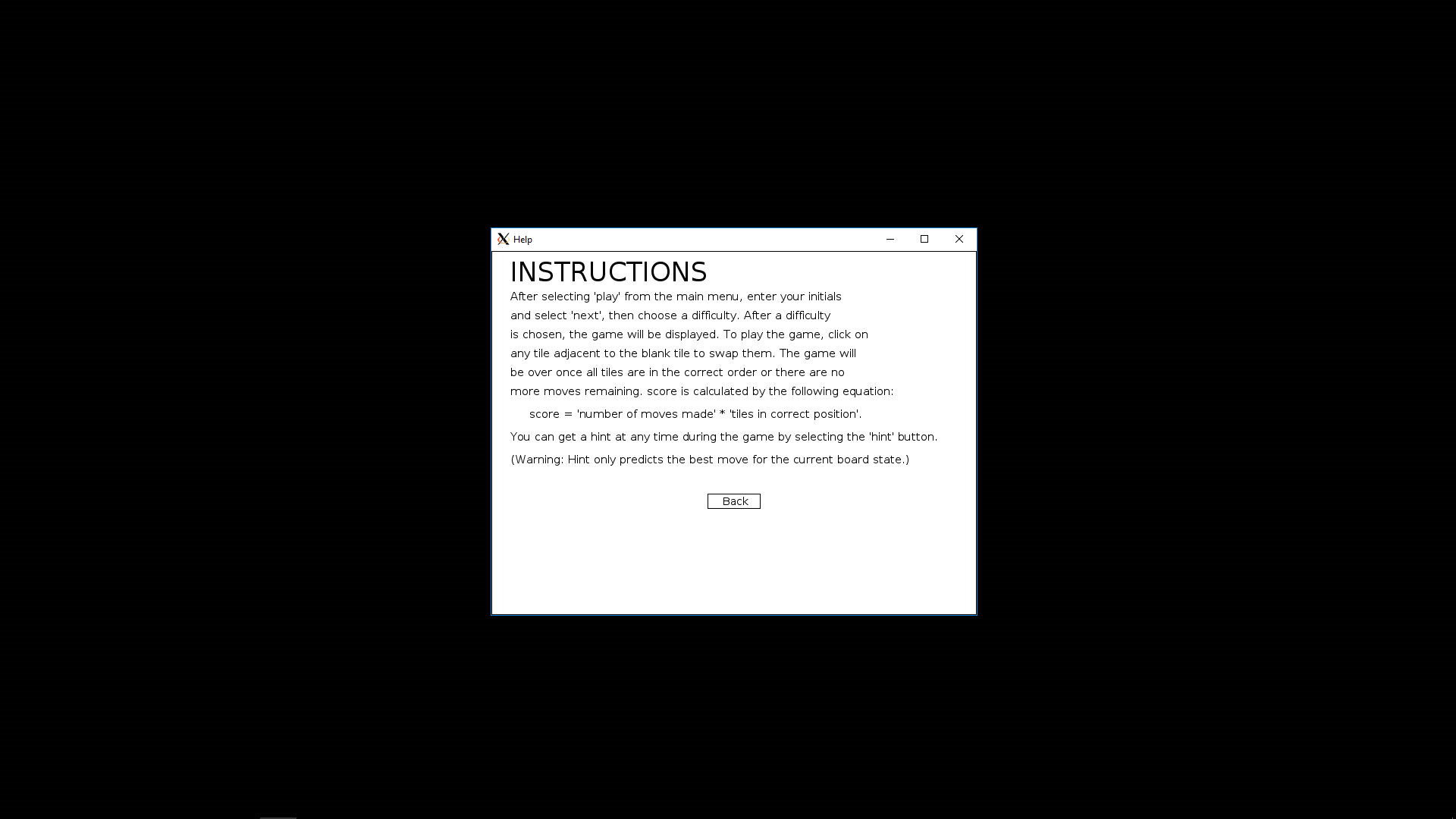
PART IV – Explanation of Approach

Each different screen (main menu, help, difficulty, initials, game, game over) is displayed through its own window, which displays on top of the previous window, and the previous window is then destroyed. In order to display different windows, a window manager was implemented in order to display and destroy windows as needed. The game itself is displayed through the Game\_window class, and a Game object is passed to it to supply the difficulty, number of moves given to the user, top scores for each difficulty, and some initial game starting conditions. The Game\_window manages information from there, updating the window with each move and implementing the core logic of the game. A score object and tile object were created to simplify reading/writing scores and implementing tile logic.

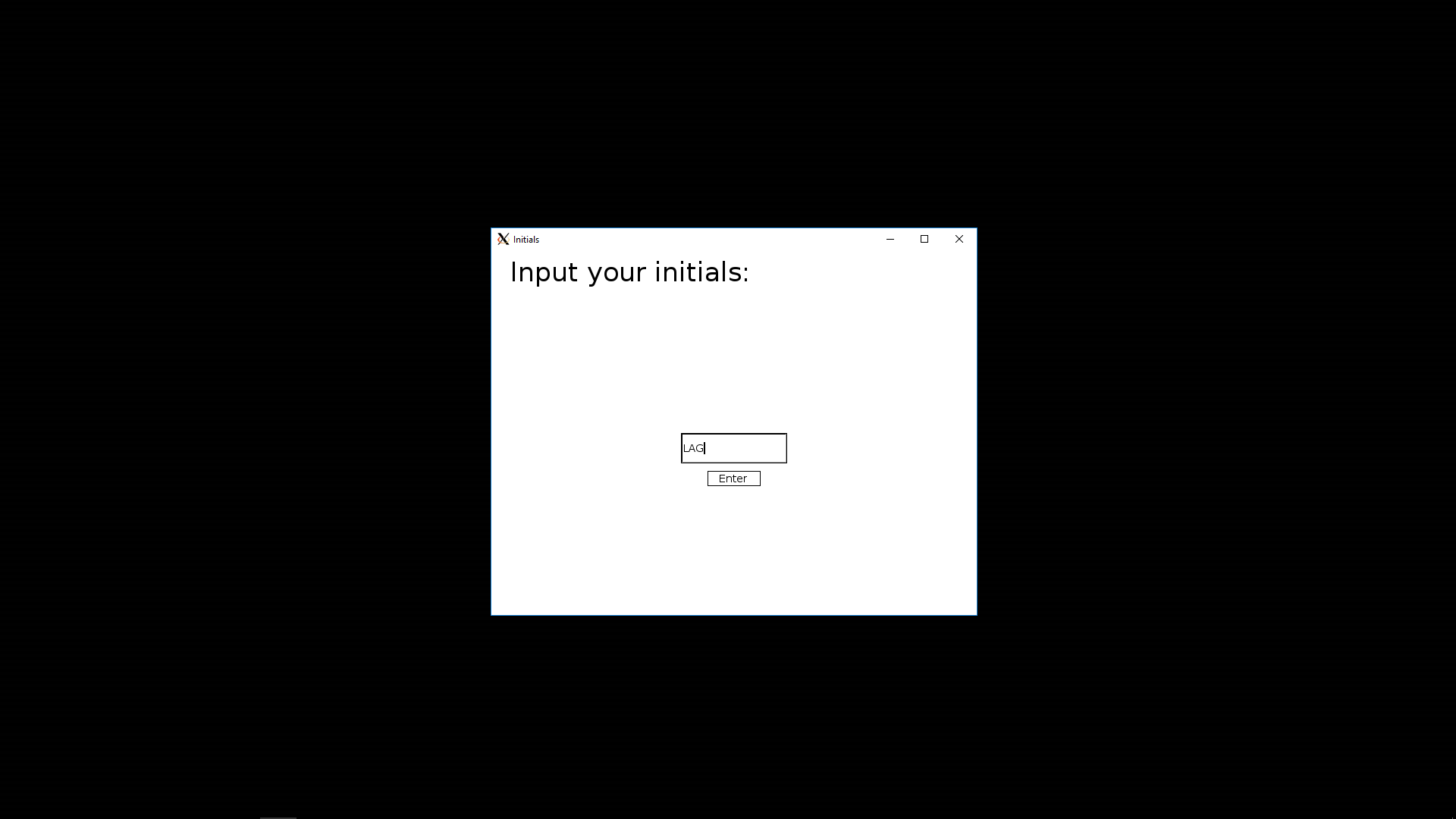
PART V – Sample run (screenshots)



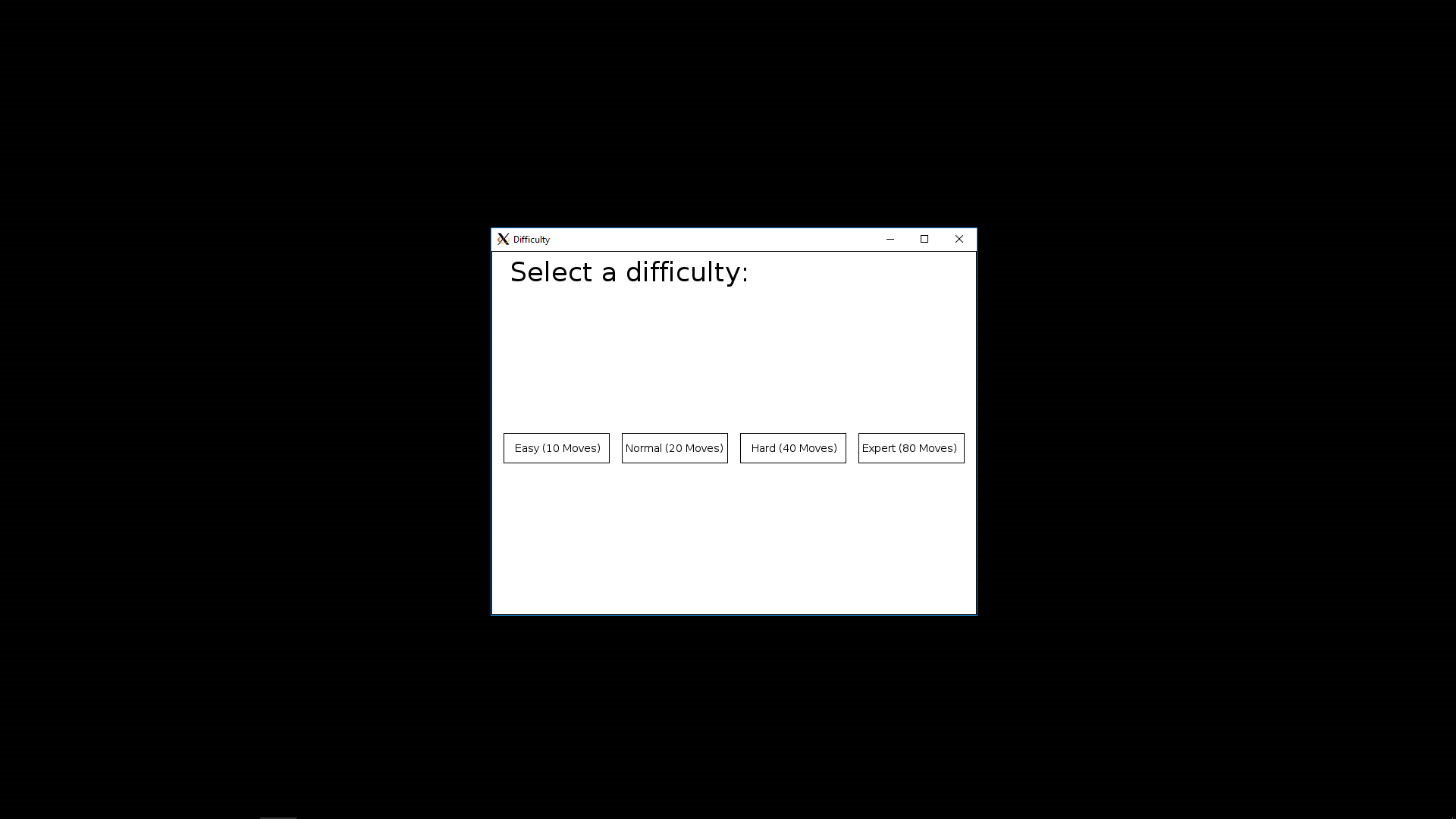
Main menu



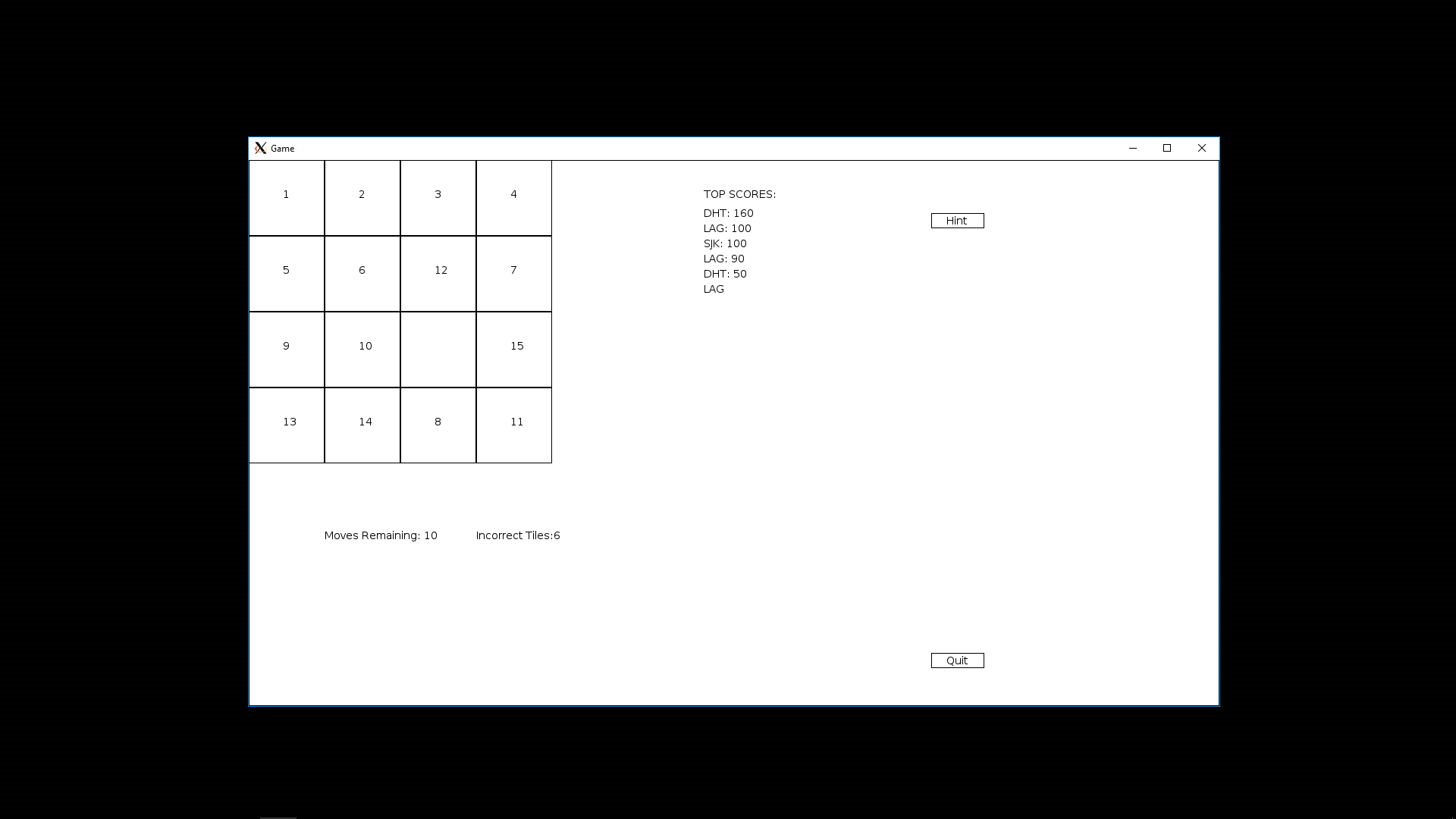
Instructions screen



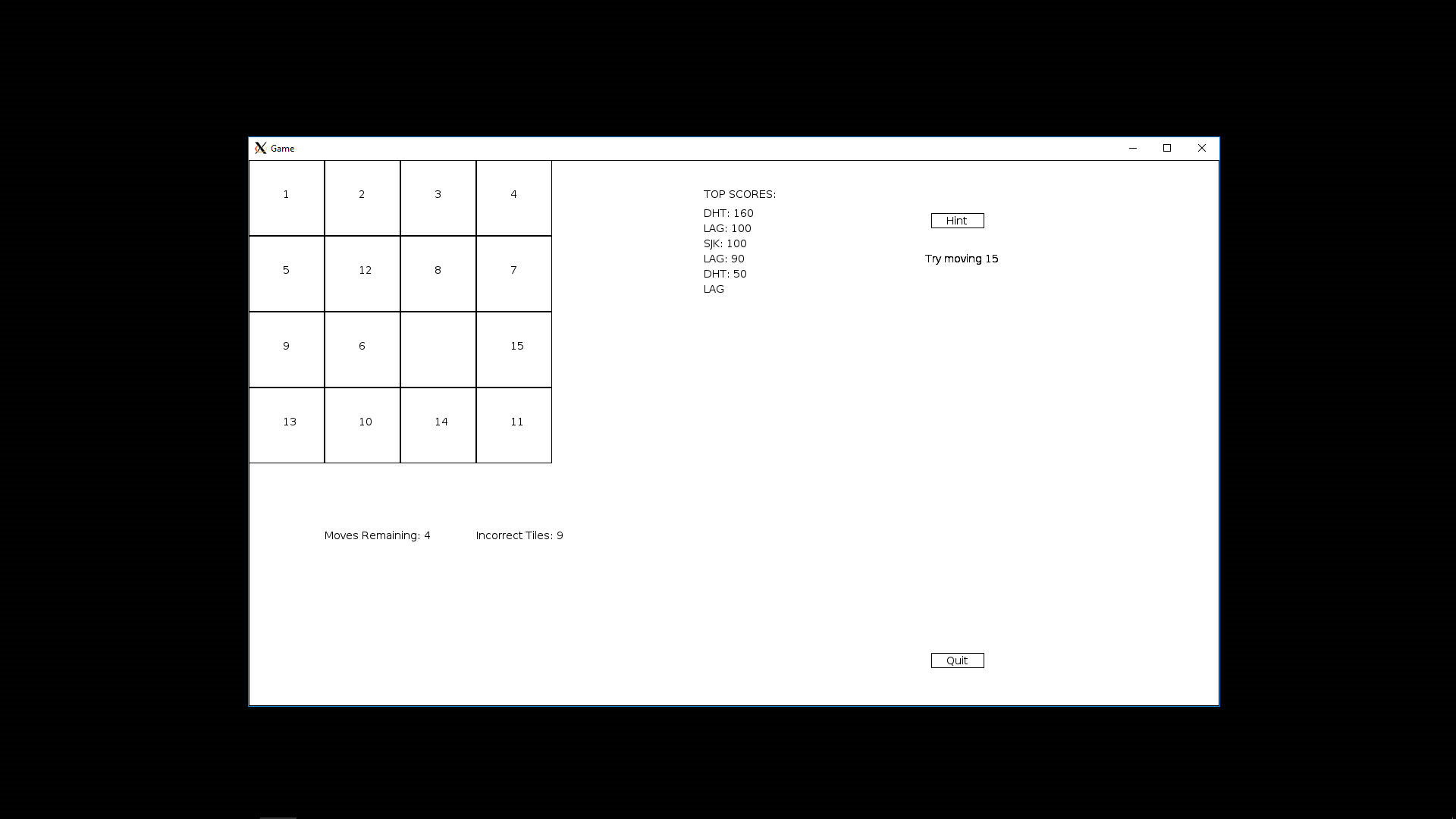
Initials screen



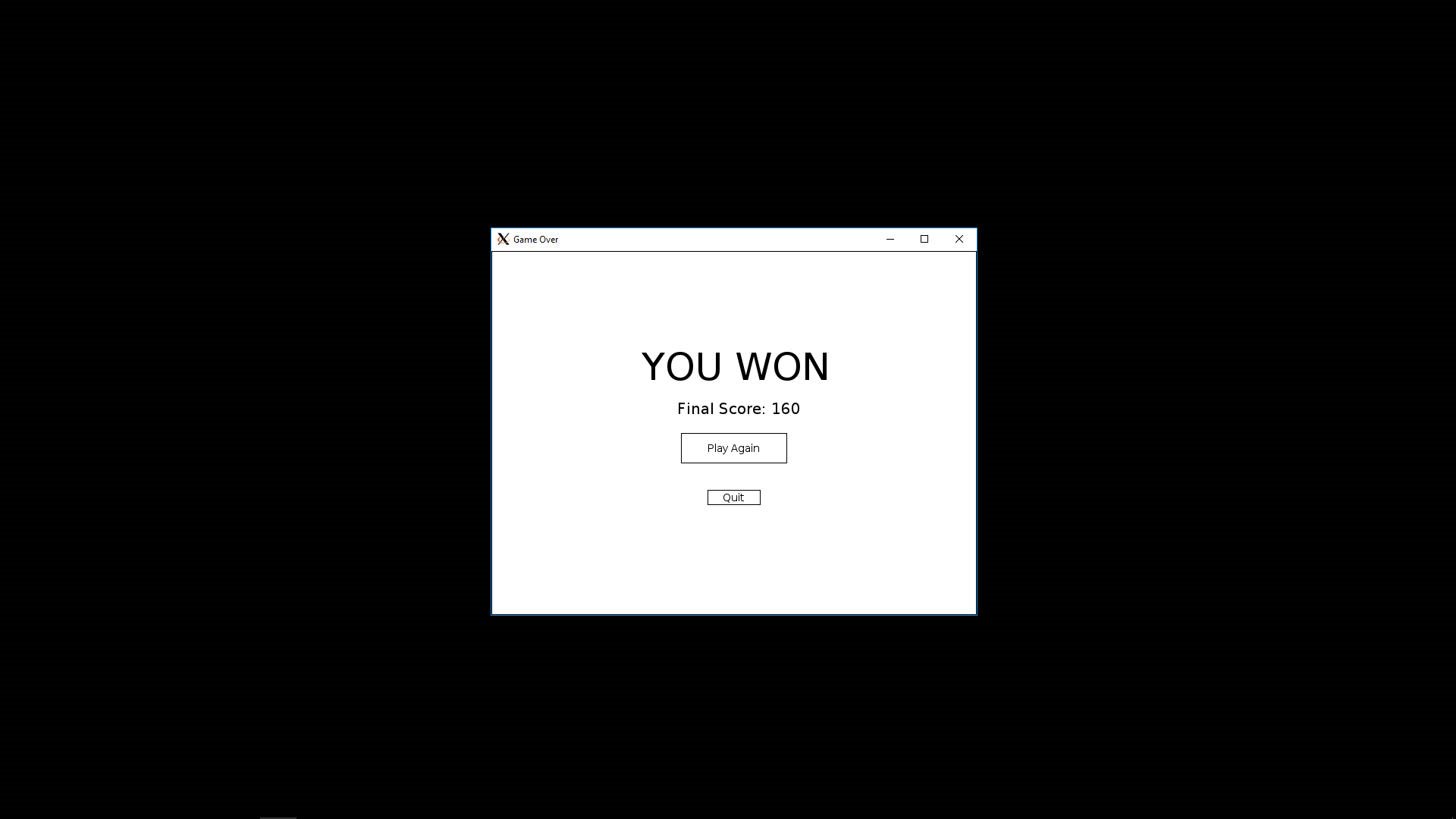
Difficulty screen



Game window



Game window with hint displayed



Game over (win)



Game over (lose)

PART VI – Results and Analysis

The game conforms to each specification, and runs more or less as expected without any known bugs which have not been fixed. The hint functionality is sub-optimal since it only recommends the best move for each particular board layout, that is to say, it doesn’t look ahead to see which move is best overall. This can lead to infinite loops where the hint button tells you to move some tile, and if you follow its recommendation and ask for another hint, it may tell you to move it back. This result is its expected behavior, however, using the Manhattan distance formula provided by the project specifications.

PART VII – Conclusions (what did you show/learn)

Learning some of the intricacies of the FLTK Graphics library, as well as the classes defined by Stroustrup, were almost essential to completing the project. We learned more about scope, user defined data types (classes and structs), and we have also gained some practical experience programming applications with a GUI. This kind of programming is different in many ways compared to console applications, and required a bit of getting used to. We were also able to practice coordinating and communicating with other group members, which was a fundamental part of the project.

PART VIII – Future Research (how it could be improved)

The game could be improved in several ways. One such way would be to design a more complex hint generator which would be able to look ahead several moves. Another way would be to redesign the window manager system to display all the different screens on one single window. Sliding tile animations could be added, or various sound effects/background music. While a minimalistic graphical style was chosen in order to meet the 24 line function requirement, graphical objects could be added relatively easily to make the game more graphically impressive, such as adding images for textures or closed polylines to simulate shadows.

PART IX – Instructions

The game should be run through PuTTY with Xming, or through a Linux terminal using the included a.out file. After running the a.out file, the main menu should appear in the center of the screen on a 1080p monitor, or offset from the center on a lower resolution screen. After selecting 'play' from the main menu, enter your initials and select 'next', then choose a difficulty. After a difficulty is chosen, the game will be displayed. To play the game, click on a tile adjacent to the blank tile to swap them. The game will be over once all tiles are in the correct order or there are no more moves remaining. score is calculated by the following equation:

score = 'number of moves made' \* 'tiles in correct position'.

You can get a hint at any time during the game by selecting the 'hint' button.

PART X – Listing of Program

//

// Window\_manager.h

// Team 4 - 'Please Compile'

//

// Created by Luke Grammer

// Copyright © 2017 Luke Grammer. All rights reserved.

#ifndef WINDOW\_MANAGER

#define WINDOW\_MANAGER

#include "Main\_window.h"

#include "Difficulty\_window.h"

#include "Help\_window.h"

#include "Initials\_window.h"

#include "Game\_window.h"

#include "Game\_over\_window.h"

class Window\_manager {

int init\_main\_window();

void init\_help();

int init\_difficulty();

void init\_initials();

int init\_game(int difficulty);

bool init\_game\_over(int score, int difficulty);

string user\_initial;

public:

Window\_manager();

void start\_game\_sequence();

};

#endif

//

// Window\_manager.cpp

// Team 4 - 'Please Compile'

//

// Created by Luke Grammer

// Copyright © 2017 Luke Grammer. All rights reserved.

#include "Window\_manager.h"

const int WINDOW\_X = 640; // Window size

const int WINDOW\_Y = 480;

const int ORIG\_X = 640; // Window origin

const int ORIG\_Y = 300;

Window\_manager::Window\_manager() {}

void Window\_manager::start\_game\_sequence() { // Runs the game

int choice;

int score;

bool game = true;

bool replay = true;

while (game) {

choice = init\_main\_window(); // Display main menu

if (choice == 1) {

init\_initials(); // Display initials screen

while (replay) {

choice = init\_difficulty(); // Display difficulty screen

score = init\_game(choice); // Display game

replay = init\_game\_over(score, choice); // Display game over

}

game = false; // Do not go back to main menu

}

else if (choice == 2) {

init\_help(); // Help screen

}

else {

game = false; // Do not go back to main menu

}

}

}

int Window\_manager::init\_main\_window() { // Display main menu

Main\_window menu{ Point{ ORIG\_X,ORIG\_Y }, WINDOW\_X, WINDOW\_Y, "Main Menu" };

menu.wait\_for\_button();

menu.quit();

return menu.get\_choice();

}

void Window\_manager::init\_help() { // Display help menu

Help\_window help{ Point{ ORIG\_X,ORIG\_Y }, WINDOW\_X, WINDOW\_Y, "Help" };

help.wait\_for\_button();

help.quit();

}

int Window\_manager::init\_difficulty() { // Display difficulty menu

Difficulty\_window difficulty{ Point{ ORIG\_X,ORIG\_Y }, WINDOW\_X, WINDOW\_Y, "Difficulty" };

difficulty.wait\_for\_button();

difficulty.quit();

return difficulty.get\_choice();

}

void Window\_manager::init\_initials() { // Display initials window

Initials\_window initials{ Point{ ORIG\_X,ORIG\_Y }, WINDOW\_X, WINDOW\_Y, "Initials" };

initials.wait\_for\_button();

user\_initial = initials.get\_initials();

initials.quit();

}

int Window\_manager::init\_game(int difficulty) { // Display game

Game game{ difficulty, user\_initial };

Game\_window game\_window(Point{ 320, 180 }, 1280, 720, "Game", game);

game\_window.wait\_for\_button();

game\_window.quit();

return game\_window.get\_final\_score();

}

bool Window\_manager::init\_game\_over(int score, int difficulty) { // Display game over

Game\_over\_window game\_over{ Point{ ORIG\_X,ORIG\_Y }, WINDOW\_X, WINDOW\_Y, "Game Over", score, difficulty};

game\_over.wait\_for\_button();

game\_over.quit();

return game\_over.get\_choice();

}

//

// User\_score.h

// Team 4 - 'Please Compile'

//

// Created by SeungJin Kim on 11/12/17.

// Copyright © 2017 SeungJin Kim. All rights reserved.

#ifndef USER\_SCORE

#define USER\_SCORE

#include "std\_lib\_facilities\_5.h"

class User\_score {

string initials;

int score;

int difficulty;

public:

User\_score(string initials, int score, int difficulty);

User\_score& swap(const User\_score &user\_score);

string get\_initials();

void set\_initials(string intials);

int get\_score();

void set\_score(int score);

int get\_difficulty();

void set\_difficulty(int difficulty);

friend ostream & operator<<(ostream& os, const User\_score & x);

friend istream & operator>> (istream& is, User\_score& z);

bool operator<(const User\_score& user\_score);

};

#endif

//

// User\_score.cpp

// Team 4 - 'Please Compile'

//

// Created by SeungJin Kim on 11/12/17.

// Copyright © 2017 SeungJin Kim. All rights reserved.

#include "User\_score.h"

User\_score::User\_score(string initials, int score, int difficulty)

: initials{ initials }, score{ score }, difficulty{ difficulty }

{

}

// Getters and Setters

string User\_score::get\_initials() {

return initials;

}

void User\_score::set\_initials(string initials) {

this->initials = initials;

}

int User\_score::get\_score() {

return score;

}

void User\_score::set\_score(int score) {

this->score = score;

}

int User\_score::get\_difficulty() {

return difficulty;

}

void User\_score::set\_difficulty(int difficulty) {

this->difficulty = difficulty;

}

// Swaps initials/score

User\_score& User\_score::swap(const User\_score &user\_score) {

// Swaps initials and score

this->initials = user\_score.initials;

this->score = user\_score.score;

return \*this;

}

// Operator definitions

ostream& operator<<(ostream & os, const User\_score & x) {

return os << "Difficulty: " << x.difficulty << " User\_Initial: " << x.initials << " Score: " << x.score;

}

istream& operator>>(istream & is, User\_score & z) {

int difficulty, score;

string initial, d, i, s;

is >> d >> difficulty >> i >> initial >> s >> score;

z = User\_score(initial, score, difficulty);

return is;

}

bool User\_score::operator<(const User\_score& user\_score) {

if (this->score < user\_score.score) return true;

else return false;

}

//

// Tile.h

// Team 4 - 'Please Compile'

//

// Created by SeungJin Kim on 11/9/17.

// Copyright © 2017 SeungJin Kim. All rights reserved.

#ifndef TILE

#define TILE

#include "GUI.h"

using namespace Graph\_lib;

class Tile : public Graph\_lib::Button {

int number;

bool space;

string title;

public:

Tile(Point xy, int wh, const string& title, int number, bool has\_space, Callback cb);

bool has\_space();

void set\_space(bool);

int get\_number();

void set\_number(int n);

string get\_title();

void set\_title(string);

Tile& operator= (const Tile &tile);

};

#endif

//

// Tile.cpp

// Team 4 - 'Please Compile'

//

// Created by SeungJin Kim on 11/9/17.

// Copyright © 2017 SeungJin Kim. All rights reserved.

//

#include "Tile.h"

Tile::Tile(Point xy, int wh, const string& title, int number, bool has\_space, Callback cb)

: Button{ xy,wh,wh,title,cb }, number{ number }, space{ has\_space }, title{ title }

{

}

// Getters and Setters

bool Tile::has\_space() {

return space;

}

void Tile::set\_space(bool x) {

space = x;

}

int Tile::get\_number() {

return number;

cout << number;

}

void Tile::set\_number(int n) {

number = n;

}

string Tile::get\_title() {

return title;

}

void Tile::set\_title(string n) {

this->title = n;

}

// Operator definition

Tile& Tile::operator=(const Tile &tile) {

this->loc.x = tile.loc.x;

this->loc.y = tile.loc.y;

this->height = tile.height;

this->width = tile.width;

this->title = tile.title;

this->number = tile.number;

this->space = tile.space;

this->label = tile.title;

return \*this;

}

//

// Main\_window.h

// Team 4 - 'Please Compile'

//

// Created by Luke Grammer

// Copyright © 2017 Luke Grammer. All rights reserved.

#ifndef MAIN\_WINDOW

#define MAIN\_WINDOW

#include "Simple\_window.h"

class Main\_window : Graph\_lib::Window {

Rectangle background;

Text splash\_text1;

Text splash\_text2;

Button start\_button;

Button help\_button;

Button quit\_button;

Rectangle start\_button\_cover;

Rectangle help\_button\_cover;

Rectangle quit\_button\_cover;

Text start\_button\_text;

Text help\_button\_text;

Text quit\_button\_text;

bool button\_pushed;

int selection;

void stylize\_objects();

void attach\_objects();

public:

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

void wait\_for\_button();

int change\_state(int choice);

int get\_choice() { return selection; }

void quit();

};

#endif

//

// Main\_window.cpp

// Team 4 - 'Please Compile'

//

// Created by Luke Grammer

// Copyright © 2017 Luke Grammer. All rights reserved.

#include "Main\_window.h"

#include "Graph.h"

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

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

background{ Point{0, 0}, Point{x\_max(), y\_max()} },

splash\_text1{ Point{ 25, 40 }, "PLEASE COMPILE - THE 16 GAME" },

splash\_text2{ Point{ 25, 65 }, "By Team 4 - Luke Grammer, Daniel Hain-Trevino, and Seungjin Kim" },

start\_button{ Point{ x\_max() / 2 - 70, y\_max() / 2 }, 140, 40, "Play",

[](Address, Address pw) {reference\_to<Main\_window>(pw).change\_state(1); } },

help\_button{ Point{ x\_max() / 2 - 35, y\_max() / 2 + 80 }, 70, 20, "Help",

[](Address, Address pw) {reference\_to<Main\_window>(pw).change\_state(2); } },

quit\_button{ Point{ x\_max() / 2 - 35, y\_max() / 2 + 140 }, 70, 20, "Quit",

[](Address, Address pw) {reference\_to<Main\_window>(pw).change\_state(0); } }, // C++11 features (lambda expressions)

start\_button\_cover{ Point{ x\_max() / 2 - 70, y\_max() / 2 }, Point{ x\_max() / 2 - 70 + 140, y\_max() / 2 + 40} },

help\_button\_cover{ Point{ x\_max() / 2 - 35, y\_max() / 2 + 80 }, Point{ x\_max() / 2 - 35 + 70, y\_max() / 2 + 80 + 20 } },

quit\_button\_cover{ Point{ x\_max() / 2 - 35, y\_max() / 2 + 140 }, Point{ x\_max() / 2 - 35 + 70, y\_max() / 2 + 140 + 20 } },

start\_button\_text{ Point{ x\_max() / 2 - 70 + 55, y\_max() / 2 + 25}, "Play" },

help\_button\_text{ Point{ x\_max() / 2 - 35 + 20, y\_max() / 2 + 80 + 15 }, "Help" },

quit\_button\_text{ Point{ x\_max() / 2 - 35 + 20, y\_max() / 2 + 140 + 15}, "Quit" },

button\_pushed{ false },

selection{ 0 }

{

stylize\_objects();

attach\_objects();

}

void Main\_window::stylize\_objects() {

// Changes appearance of objects

background.set\_fill\_color(Color::white);

splash\_text1.set\_font\_size(35);

splash\_text2.set\_font\_size(15);

}

void Main\_window::attach\_objects() {

// Attaches all objects to window

attach(background);

attach(splash\_text1);

attach(splash\_text2);

attach(start\_button);

attach(help\_button);

attach(quit\_button);

attach(start\_button\_cover);

attach(help\_button\_cover);

attach(quit\_button\_cover);

attach(start\_button\_text);

attach(help\_button\_text);

attach(quit\_button\_text);

}

void Main\_window::quit() {

hide(); // Closes window

}

int Main\_window::change\_state(int choice) {

// Detects button push, and returns which button was pushed

button\_pushed = true;

selection = choice;

return choice;

}

void Main\_window::wait\_for\_button() {

// Waits for button push

show();

while (!button\_pushed) {

Fl::wait();

}

}

//

// Initials\_window.h

// Team 4 - 'Please Compile'

//

// Created by Daniel Hain-Trevino

// Copyright © 2017 Daniel Hain-Trevino. All rights reserved.

#ifndef INITIALS\_WINDOW

#define INITIALS\_WINDOW

#include "Simple\_window.h"

class Initials\_window : Graph\_lib::Window {

Rectangle background\_top;

Rectangle background\_left;

Rectangle background\_bottom;

Rectangle background\_right;

Text initials\_text;

In\_box initials\_input;

Button initials\_button;

Rectangle initials\_button\_cover;

Rectangle initials\_input\_cover;

Text initials\_button\_text;

string initials\_string;

bool button\_pushed;

void stylize\_objects();

void attach\_objects();

void store\_init();

public:

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

void wait\_for\_button();

string get\_initials() { return initials\_string; }

void quit();

};

#endif

//

// Initials\_window.cpp

// Team 4 - 'Please Compile'

//

// Created by Daniel Hain-Trevino

// Copyright © 2017 Daniel Hain-Trevino. All rights reserved.

#include "Initials\_window.h"

#include "Graph.h"

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

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

background\_top{ Point{0, 0}, Point{640, y\_max() / 2} },

background\_left{ Point{0, 0}, Point{x\_max() / 2 - 70, 480} },

background\_bottom{ Point{0, y\_max() / 2 + 40}, Point{640, 480} },

background\_right{ Point{x\_max() / 2 + 70, 0}, Point{640, 480} },

initials\_text{ Point{ 25, 40 }, "Input your initials:" },

initials\_input{ Point{ x\_max() / 2 - 70, y\_max() / 2 }, 140, 40, "" },

initials\_button{ Point { x\_max() / 2 - 35, y\_max() / 2 + 50 }, 70, 20, "Enter", [](Address, Address pw) {reference\_to<Initials\_window>(pw).store\_init(); } }, // C++11 Feature (lamba expressions)

initials\_button\_cover{ Point { x\_max() / 2 - 35, y\_max() / 2 + 50 }, Point{ x\_max() / 2 + 35, y\_max() / 2 + 70} },

initials\_input\_cover{ Point { x\_max() / 2 - 70, y\_max() / 2 }, Point{ x\_max() / 2 + 70, y\_max() / 2 + 40} },

initials\_button\_text{ Point{ x\_max() / 2 - 35 + 15, y\_max() / 2 + 65 }, "Enter" },

initials\_string{ "" },

button\_pushed{ false }

{

stylize\_objects();

attach\_objects();

}

void Initials\_window::stylize\_objects() { // Changes appearance of objects

background\_top.set\_fill\_color(Color::white);

background\_left.set\_fill\_color(Color::white);

background\_bottom.set\_fill\_color(Color::white);

background\_right.set\_fill\_color(Color::white);

background\_top.set\_color(Color::white);

background\_left.set\_color(Color::white);

background\_bottom.set\_color(Color::white);

background\_right.set\_color(Color::white);

initials\_text.set\_font\_size(35);

}

void Initials\_window::attach\_objects() { // Attaches objects

attach(background\_top);

attach(background\_left);

attach(background\_bottom);

attach(background\_right);

attach(initials\_text);

attach(initials\_input);

attach(initials\_button);

attach(initials\_button\_cover);

attach(initials\_input\_cover);

attach(initials\_button\_text);

}

void Initials\_window::store\_init() {

// Stores initials & registers button push

button\_pushed = true;

initials\_string = initials\_input.get\_string();

}

void Initials\_window::quit() { // Close window

hide();

}

void Initials\_window::wait\_for\_button() { // Wait for button push

show();

while (!button\_pushed) {

Fl::wait();

}

}

//

// Hint.h

// Team 4 - 'Please Compile'

//

// Created by Luke Grammer

// Copyright © 2017 Luke Grammer. All rights reserved.

#ifndef HINT

#define HINT

#include "Simple\_window.h"

class Hint {

int simulate\_valid\_moves(vector<vector<int>> board);

void find\_16(int& x\_loc, int& y\_loc, const vector<vector<int>>& board);

vector<vector<int>> swap(int num1, int num2, vector<vector<int>> board);

int find\_distance(const vector<vector<int>>& board);

public:

Hint();

string create\_hint(vector<vector<int>> board);

};

#endif

//

// Hint.cpp

// Team 4 - 'Please Compile'

//

// Created by Luke Grammer

// Copyright © 2017 Luke Grammer. All rights reserved.

#include "Hint.h"

Hint::Hint()

{

}

string Hint::create\_hint(vector<vector<int>> board) {

// Finds manhattan dist. of all possible moves and recommends lowest

return "Try moving " + to\_string(simulate\_valid\_moves(board));

}

int Hint::simulate\_valid\_moves(vector<vector<int>> board) {

// Simulates all valid moves, and returns

// value corresponding to move which causes the smallest manhattan dist.

int x\_loc = 0, y\_loc = 0, d1 = 1000000, d2 = d1, d3 = d1, d4 = d1;

find\_16(x\_loc, y\_loc, board); // Find coordinates of 16 tile

if (x\_loc + 1 < 4) // If swapping with tile below is valid

d1 = find\_distance(swap(board[x\_loc + 1][y\_loc], board[x\_loc][y\_loc], board));

if (x\_loc - 1 > 0) // If swapping with tile above is valid

d2 = find\_distance(swap(board[x\_loc - 1][y\_loc], board[x\_loc][y\_loc], board));

if (y\_loc + 1 < 4) // If swapping with tile left is valid

d3 = find\_distance(swap(board[x\_loc][y\_loc + 1], board[x\_loc][y\_loc], board));

if (y\_loc - 1 > 0) // If swapping with tile right is valid

d4 = find\_distance(swap(board[x\_loc][y\_loc - 1], board[x\_loc][y\_loc], board));

if (min(d1, min(d2, min(d3, d4))) == d1) return board[x\_loc + 1][y\_loc];

else if (min(d1, min(d2, min(d3, d4))) == d2) return board[x\_loc - 1][y\_loc];

else if (min(d1, min(d2, min(d3, d4))) == d3) return board[x\_loc][y\_loc + 1];

else return board[x\_loc][y\_loc - 1];

}

void Hint::find\_16(int& x\_loc, int& y\_loc, const vector<vector<int>>& board) {

// Finds tile on board with value 16 and gives coords to argument

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

for (int j = 0; j < 4; ++j) {

if (board[i][j] == 16) {

x\_loc = i;

y\_loc = j;

}

}

}

}

vector<vector<int>> Hint::swap(int num1, int num2, vector<vector<int>> board) {

// Swaps two numbers in a 2-d vector of ints (a vector of vectors of ints)

int num1\_x, num1\_y, num2\_x, num2\_y;

for (unsigned i = 0; i < 4; ++i) {

for (unsigned j = 0; j < 4; ++j) {

if (board[i][j] == num1) {

num1\_x = i; // Find coords of first tile

num1\_y = j;

}

else if (board[i][j] == num2) {

num2\_x = i; // Find coords of second tile

num2\_y = j;

}

}

}

int temp = board[num1\_x][num1\_y]; // Swap with temporary variable

board[num1\_x][num1\_y] = board[num2\_x][num2\_y];

board[num2\_x][num2\_y] = temp;

return board; // Return new board

}

int Hint::find\_distance(const vector<vector<int>>& board) {

// Finds manhattan distance of a board (a vector of vectors of integers 1-16)

int m\_dist = 0;

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

for (int j = 0; j < 4; ++j) {

int targ\_x = ((board[i][j] - 1) / 4);

int targ\_y = (board[i][j] - 1) % 4;

int dist\_x = abs(i - targ\_x);

int dist\_y = abs(j - targ\_y);

m\_dist += dist\_x + dist\_y;

}

}

return m\_dist;

}

//

// Help\_window.h

// Team 4 - 'Please Compile'

//

// Created by Luke Grammer

// Copyright © 2017 Luke Grammer. All rights reserved.

#ifndef HELP\_WINDOW

#define HELP\_WINDOW

#include "Simple\_window.h"

class Help\_window : Graph\_lib::Window {

Rectangle background;

Text help\_text\_main;

Vector\_ref<Text> help\_text;

Button back\_button;

Rectangle back\_button\_cover;

Text back\_button\_text;

bool button\_pushed;

public:

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

void stylize\_objects();

void attach\_objects();

void change\_state();

void quit();

void wait\_for\_button();

};

#endif

//

// Help\_window.cpp

// Team 4 - 'Please Compile'

//

// Created by Luke Grammer

// Copyright © 2017 Luke Grammer. All rights reserved.

#include "Help\_window.h"

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

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

background{ Point{ 0, 0 }, Point{ x\_max(), y\_max() } },

help\_text\_main{ Point{ 25, 40 }, "INSTRUCTIONS" },

back\_button{ Point{ x\_max() / 2 - 35, y\_max() / 2 + 80 }, 70, 20, "Back", [](Address, Address pw) {reference\_to<Help\_window>(pw).change\_state(); } },

back\_button\_cover{ Point{ x\_max() / 2 - 35, y\_max() / 2 + 80 }, Point{ x\_max() / 2 - 35 + 70, y\_max() / 2 + 80 + 20 } },

back\_button\_text{ Point{ x\_max() / 2 - 35 + 20, y\_max() / 2 + 80 + 15 }, "Back" },

button\_pushed{ false }

{

help\_text.push\_back(new Text(Point{ 25, 65 }, "After selecting 'play' from the main menu, enter your initials")),

help\_text.push\_back(new Text(Point{ 25, 90 }, "and select 'next', then choose a difficulty. After a difficulty")),

help\_text.push\_back(new Text(Point{ 25, 115 }, "is chosen, the game will be displayed. To play the game, click on")),

help\_text.push\_back(new Text(Point{ 25, 140 }, "any tile adjacent to the blank tile to swap them. The game will")),

help\_text.push\_back(new Text(Point{ 25, 165 }, "be over once all tiles are in the correct order or there are no")),

help\_text.push\_back(new Text(Point{ 25, 190 }, "more moves remaining. score is calculated by the following equation:")),

help\_text.push\_back(new Text(Point{ 25, 220 }, " score = 'number of moves made' \* 'tiles in correct position'.")),

help\_text.push\_back(new Text(Point{ 25, 250 }, "You can get a hint at any time during the game by selecting the 'hint' button.")),

help\_text.push\_back(new Text(Point{ 25, 280 }, "(Warning: Hint only predicts the best move for the current board state.)")),

stylize\_objects();

attach\_objects();

}

void Help\_window::stylize\_objects() { // Change appearance of screen objects

background.set\_fill\_color(Color::white);

help\_text\_main.set\_font\_size(35);

for (unsigned i = 0; i < help\_text.size(); ++i)

help\_text[i].set\_font\_size(15);

}

void Help\_window::attach\_objects() { // Attach objects to window

attach(background);

attach(help\_text\_main);

attach(back\_button);

attach(back\_button\_cover);

attach(back\_button\_text);

for (unsigned i = 0; i < help\_text.size(); ++i)

attach(help\_text[i]);

}

void Help\_window::quit() { // Close window

hide();

}

void Help\_window::change\_state() { // Detects button push

button\_pushed = true;

}

void Help\_window::wait\_for\_button() { // Wait for button push

show();

while (!button\_pushed) {

Fl::wait();

}

}

//

// Game\_window.h

// Team 4 - 'Please Compile'

//

// Created by SeungJin Kim on 11/10/17.

// Copyright © 2017 SeungJin Kim. All rights reserved.

#ifndef GAME\_WINDOW

#define GAME\_WINDOW

#include "Tile.h"

#include "Game.h"

#include "Hint.h"

#include "Simple\_window.h"

#include "FL/Fl\_Button.H"

class Game\_window : Graph\_lib::Window {

Vector\_ref<Tile> board;

Vector\_ref<Text> numbers;

Vector\_ref<Text> scores;

Game game;

Rectangle background;

Button hint\_button;

Text hint\_button\_text;

Button quit\_button;

Text quit\_button\_text;

Text hint\_text;

Text moves\_left;

Text tiles\_needed\_move;

vector<User\_score> top\_scores;

bool button\_pushed;

Vector\_ref<Rectangle> board\_cover;

vector<vector<int>> matrix;

int empty\_index;

int final\_score;

void change\_state();

void stylize\_objects();

void attach\_objects();

void load\_game(int difficulty);

void load\_board(int difficulty);

void get\_matrix(int rand\_num);

void load\_tiles();

void load\_scores();

static void cb\_tile(Address p, Address pw);

void swap\_tile(Tile& tile1, Tile& tile2, Text& tile1\_label, Text& tile2\_label);

void update\_information();

void export\_scores();

void write\_to\_file();

void set\_moveable(int index, bool a);

void find\_incorrect\_tiles();

void create\_hint\_text();

public:

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

void move\_button(Address);

void wait\_for\_button();

int get\_final\_score();

void quit();

};

#endif

//

// GameWindow.cpp

// Team 4 - 'Please Compile'

//

// Created by SeungJin Kim on 11/10/17.

// Copyright © 2017 SeungJin Kim. All rights reserved.

#include "Game\_window.h"

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

:Window{ xy,w,h,title }, board{ 0 }, numbers{ 0 }, scores{ 0 }, game{ game },

background{ Point{ 0, 0 }, Point{ x\_max(), y\_max() } },

hint\_button{ Point{ 900, 70 }, 70, 20, "Hint",

[](Address, Address pw) {reference\_to<Game\_window>(pw).create\_hint\_text(); } },

hint\_button\_text(Point(920, 85), "Hint"),

quit\_button{ Point{ 900, 650 }, 70, 20, "Quit",

[](Address, Address pw) {reference\_to<Game\_window>(pw).change\_state(); } }, // C++11 Feature (lambda expressions)

quit\_button\_text(Point(920, 665), "Quit"),

hint\_text{ Point{ 892, 135 }, "" },

moves\_left(Point(100, 500), "Moves Remaining: " + to\_string(game.get\_moves\_left())),

tiles\_needed\_move(Point(300, 500), "Incorrect Tiles:" + to\_string(game.get\_num\_tiles\_incorrect\_position())),

top\_scores{ game.get\_top\_scores() },

button\_pushed{ false }

{

load\_game(game.get\_difficulty());

stylize\_objects();

attach\_objects();

}

void Game\_window::quit() { // Closes window

hide();

}

void Game\_window::wait\_for\_button() { // Waits for non-tile button push

show();

while (!button\_pushed) {

Fl::wait();

}

}

void Game\_window::change\_state() { // Registers non-tile button push

button\_pushed = true;

}

void Game\_window::stylize\_objects() { // Changes appearance of screen objects

background.set\_fill\_color(Color::white);

for (unsigned i = 0; i < board.size(); ++i) {

board\_cover[i].set\_fill\_color(Color::white);

}

board\_cover[board.size()].set\_fill\_color(Color::white);

board\_cover[board.size() + 1].set\_fill\_color(Color::white);

}

void Game\_window::attach\_objects() { // Attaches screen objects

attach(background);

for (unsigned i = 0; i < board.size(); ++i) {

attach(board[i]);

attach(board\_cover[i]);

attach(numbers[i]);

}

for (unsigned i = 0; i < scores.size(); ++i) {

attach(scores[i]);

}

attach(hint\_button);

attach(board\_cover[board.size()]); // Hint button cover

attach(hint\_button\_text);

attach(quit\_button);

attach(board\_cover[board.size() + 1]); // Quit button cover

attach(quit\_button\_text);

attach(moves\_left);

attach(tiles\_needed\_move);

}

void Game\_window::load\_game(int difficulty) {

// Calls load board & load scores, locates special tile, and updates incorrect tile indicator

load\_board(difficulty);

load\_scores();

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

if (board[i].get\_title() == "16") {

empty\_index = i;

set\_moveable(i, true);

}

}

set\_moveable(empty\_index, true);

find\_incorrect\_tiles();

tiles\_needed\_move.set\_label("Incorrect Tiles: " + to\_string(game.get\_num\_tiles\_incorrect\_position()));

}

// Chooses a random number based on difficulty, and calls get matrix & load tiles

void Game\_window::load\_board(int difficulty) {

int rand\_num = difficulty;

if (rand\_num == 1) { rand\_num = rand() % (4 - 1 + 1) + 1; }

else if (rand\_num == 2) { rand\_num = rand() % (8 - 5 + 1) + 5; }

else if (rand\_num == 3) { rand\_num = rand() % (12 - 9 + 1) + 9; }

else if (rand\_num == 4) { rand\_num = rand() % (16 - 13 + 1) + 13; }

get\_matrix(rand\_num);

load\_tiles();

}

// Gives board starting positions based on the random number

void Game\_window::get\_matrix(int rand\_num) {

switch (rand\_num) {

// Easy starting positions (10 moves)

case 1: matrix = { {1,2,3,4},{5,6,12,7},{9,10,16,15},{13,14,8,11} }; break;

case 2: matrix = { {1,6,2,4},{5,10,3,8},{16,14,7,12},{9,13,11,15} }; break;

case 3: matrix = { {5,1,2,3},{9,6,7,4},{13,10,11,8},{14,16,15,12} }; break;

case 4: matrix = { {1,2,7,3},{5,16,11,4},{9,6,15,8},{13,10,14,12} }; break;

// Normal starting positions (20 moves)

case 5: matrix = { {1,6,3,4},{5,16,2,11},{9,10,14,7},{13,15,12,8} }; break;

case 6: matrix = { {2,5,3,4},{13,16,6,8},{10,1,9,12},{14,11,7,15} }; break;

case 7: matrix = { {5,1,2,4},{13,9,3,8},{15,6,7,11},{14,16,10,12} }; break;

case 8: matrix = { {11,2,16,4},{1,5,3,8},{10,7,14,12},{9,6,13,15} }; break;

// Hard starting positions (40 moves)

case 9: matrix = { {4,8,12,15},{3,16,7,11},{9,2,1,10},{13,14,5,6} }; break;

case 10: matrix = { {9,5,1,4},{10,13,7,3},{15,6,14,8},{2,11,12,16} }; break;

case 11: matrix = { {1,3,7,8},{4,13,5,10},{2,9,6,15},{14,16,12,11} }; break;

case 12: matrix = { {15,5,6,2},{1,16,9,7},{10,13,8,3},{14,12,4,11} }; break;

// Expert starting positions (80 moves)

case 13: matrix = { {16,12,11,13},{15,14,10,9},{3,7,6,2},{4,8,5,1} }; break;

case 14: matrix = { {16,12,9,13},{15,11,10,14},{3,7,2,5},{4,8,6,1} }; break;

case 15: matrix = { {16,15,9,13},{11,12,10,14},{3,7,6,2},{4,8,5,1} }; break;

case 16: matrix = { {16,12,14,13},{15,11,9,10},{8,3,6,2},{4,7,5,1} }; break;

}

}

// Adds tiles and backgrounds

void Game\_window::load\_tiles() {

for (auto i = 0; i < 4; ++i) { // C++11 feature (auto variables)

for (auto j = 0; j < 4; ++j) {

board.push\_back(new Tile(Point(j \* 100, i \* 100), 100,

to\_string(matrix[i][j]), matrix[i][j], false, cb\_tile));

board\_cover.push\_back(new Rectangle{ Point(j \* 100, i \* 100),

Point(j \* 100 + 100, i \* 100 + 100) });

numbers.push\_back(new Text(Point(j \* 100 + 45, i \* 100 + 50),

(board[board.size() - 1].get\_title() != "16") ? board[board.size() - 1].get\_title() : ""));

}

}

board\_cover.push\_back(new Rectangle{ Point{ 900, 70 }, Point{ 970, 90 } });

board\_cover.push\_back(new Rectangle{ Point{ 900, 650 }, Point{ 970, 670 } });

}

// Populates scores to display on screen

void Game\_window::load\_scores() {

scores.push\_back(new Text(Point(600, 50), "TOP SCORES:"));

for (auto i = 0; i < top\_scores.size(); ++i) {

if (i == top\_scores.size() - 1) {

scores.push\_back(new Text(Point(600, i \* 20 + 75),

top\_scores[i].get\_initials()));

}

else {

scores.push\_back(new Text(Point(600, i \* 20 + 75),

top\_scores[i].get\_initials() + ": " + to\_string(top\_scores[i].get\_score())));

}

}

}

// when Tile is clicked

void Game\_window::cb\_tile(Address p, Address pw) {

reference\_to<Game\_window>(pw).move\_button(p);

}

// Performs necessary actions after tile is clicked

void Game\_window::move\_button(Address i) {

string clicked\_number(reference\_to<Fl\_Button>(i).label());

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

if (clicked\_number == board[i].get\_title() && board[i].has\_space()) {

swap\_tile(board[i], board[empty\_index], numbers[i], numbers[empty\_index]);

set\_moveable(empty\_index, false);

empty\_index = i;

set\_moveable(empty\_index, true);

update\_information();

if (game.get\_moves\_left() == 0) {

export\_scores();

change\_state();

}

redraw();

break;

}

}

}

// Swaps two tiles using a temporary variable

void Game\_window::swap\_tile(Tile& tile1, Tile& tile2, Text& tile1\_label, Text& tile2\_label) {

Tile temp\_tile(Point(0, 0), 0, "", 0, false, cb\_tile);

temp\_tile = tile1;

tile1\_label.set\_label((tile2.get\_title() != "16") ? tile2.get\_title() : "");

tile2\_label.set\_label((tile1.get\_title() != "16") ? tile1.get\_title() : "");

tile1 = tile2;

tile2 = temp\_tile;

}

// Updates board after tile swap

void Game\_window::update\_information() {

game.set\_moves\_left(game.get\_moves\_left() - 1);

moves\_left.set\_label("Moves Remaining: " + to\_string(game.get\_moves\_left()));

find\_incorrect\_tiles();

tiles\_needed\_move.set\_label("Incorrect Tiles: " + to\_string(game.get\_num\_tiles\_incorrect\_position()));

game.set\_move\_count(game.get\_move\_count() + 1);

}

// Finds top 5 final scores and calls write\_to\_file()

void Game\_window::export\_scores() {

top\_scores[(int)top\_scores.size() - 1].set\_score((16 - game.get\_num\_tiles\_incorrect\_position()) \* game.get\_move\_count());

final\_score = top\_scores[(int)top\_scores.size() - 1].get\_score();

sort(top\_scores.rbegin(), top\_scores.rend());

if (top\_scores.size() > 5) top\_scores.pop\_back();

write\_to\_file();

}

// Writes scores to file

void Game\_window::write\_to\_file() {

ofstream score\_file{ "Scores.txt" };

for (int i = 0; i < top\_scores.size(); ++i) {

score\_file << top\_scores[i] << "\r\n";

}

if (game.get\_different\_difficulties().size() > 0) {

for (int i = 0; i < game.get\_different\_difficulties().size(); ++i) {

score\_file << game.get\_different\_difficulties()[i] << "\r\n";

}

}

score\_file.close();

}

// Determines if a tile is moveable

void Game\_window::set\_moveable(int index, bool a) {

if (index <= 3)

board[index + 4].set\_space(a);

else if (index >= 12)

board[index - 4].set\_space(a);

else {

board[index + 4].set\_space(a);

board[index - 4].set\_space(a);

}

if (index % 4 == 0)

board[index + 1].set\_space(a);

else if (index % 4 == 3)

board[index - 1].set\_space(a);

else {

board[index + 1].set\_space(a);

board[index - 1].set\_space(a);

}

}

// Determines which tiles are out of order

void Game\_window::find\_incorrect\_tiles() {

int count = 0;

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

if (board[i].get\_number() != i + 1) {

++count;

}

}

game.set\_num\_tiles\_incorrect\_position(count);

}

// Displays hint message after button is clicked

void Game\_window::create\_hint\_text() {

Hint hint;

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

for (unsigned j = 0; j < 4; ++j)

matrix[i][j] = board[j + 4 \* i].get\_number();

hint\_text.set\_label(hint.create\_hint(matrix));

attach(hint\_text);

redraw();

}

// Returns final score

int Game\_window::get\_final\_score() {

return final\_score;

//

// Game\_over\_window.h

// Team 4 - 'Please Compile'

//

// Created by Daniel Hain-Trevino.

// Copyright © 2017 Daniel Hain-Trevino. All rights reserved.

#ifndef GAME\_OVER\_WINDOW

#define GAME\_OVER\_WINDOW

#include "Simple\_window.h"

class Game\_over\_window : Graph\_lib::Window {

Rectangle background;

Text score;

Text game\_over\_text;

Button play\_button;

Button quit\_button;

Rectangle play\_button\_cover;

Rectangle quit\_button\_cover;

Text play\_button\_text;

Text quit\_button\_text;

bool button\_pushed;

int selection;

int final\_score;

int difficulty;

void check\_win();

void stylize\_objects();

void attach\_objects();

public:

Game\_over\_window(Point xy, int w, int h, const string& title, int final\_score, int difficulty);

void wait\_for\_button();

bool change\_state(bool choice);

bool get\_choice() { return selection; }

void quit();

};

#endif

//

// Game\_over\_window.cpp

// Team 4 - 'Please Compile'

//

// Created by Daniel Hain-Trevino.

// Copyright © 2017 Daniel Hain-Trevino. All rights reserved.

#include "Game\_over\_window.h"

#include "Graph.h"

Game\_over\_window::Game\_over\_window(Point xy, int w, int h, const string& title, int final\_score, int difficulty)

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

background{ Point{0, 0}, Point{x\_max(), y\_max()} },

score{ Point{ x\_max() / 2 - 75, y\_max() / 4 + 95 }, "Final Score: " + to\_string(final\_score) },

game\_over\_text{ Point{ x\_max() / 4 + 10, y\_max() / 4 + 50 }, "GAME OVER" },

play\_button{ Point{ x\_max() / 2 - 70, y\_max() / 2 }, 140, 40, "Play Again",

[](Address, Address pw) {reference\_to<Game\_over\_window>(pw).change\_state(true); } },

quit\_button{ Point{ x\_max() / 2 - 35, y\_max() / 2 + 75 }, 70, 20, "Quit",

[](Address, Address pw) {reference\_to<Game\_over\_window>(pw).change\_state(false); } }, // C++11 feature (lambda expressions)

play\_button\_cover{ Point{ x\_max() / 2 - 70, y\_max() / 2 }, Point{ x\_max() / 2 - 70 + 140, y\_max() / 2 + 40} },

quit\_button\_cover{ Point{ x\_max() / 2 - 35, y\_max() / 2 + 75 }, Point{ x\_max() / 2 - 35 + 70, y\_max() / 2 + 75 + 20 } },

play\_button\_text{ Point{ x\_max() / 2 - 70 + 35, y\_max() / 2 + 25}, "Play Again" },

quit\_button\_text{ Point{ x\_max() / 2 - 35 + 20, y\_max() / 2 + 75 + 15}, "Quit" },

button\_pushed{ false },

selection{ false },

final\_score{ final\_score },

difficulty{ difficulty }

{

check\_win();

stylize\_objects();

attach\_objects();

}

void Game\_over\_window::stylize\_objects() {

// Changes appearance of screen objects

background.set\_fill\_color(Color::white);

game\_over\_text.set\_font\_size(50);

score.set\_font\_size(20);

}

void Game\_over\_window::attach\_objects() { // Attaches objects to screen

attach(background);

attach(game\_over\_text);

attach(score);

attach(play\_button);

attach(quit\_button);

attach(play\_button\_cover);

attach(quit\_button\_cover);

attach(play\_button\_text);

attach(quit\_button\_text);

}

void Game\_over\_window::check\_win() { // Checks if game was won

switch (final\_score) {

case 160: if (difficulty == 1) { // If score is max for difficulty

game\_over\_text.set\_label("YOU WON"); // Change to win-text

game\_over\_text.move(28, 0); // Shift text to left a bit

}

break;

case 320: if (difficulty == 2) {

game\_over\_text.set\_label("YOU WON");

game\_over\_text.move(28, 0);

}

break;

case 640: if (difficulty == 3) {

game\_over\_text.set\_label("YOU WON");

game\_over\_text.move(28, 0);

}

break;

case 1280: if (difficulty == 4) {

game\_over\_text.set\_label("YOU WON");

game\_over\_text.move(28, 0);

}

break;

}

}

void Game\_over\_window::quit() { // Closes window

hide();

}

bool Game\_over\_window::change\_state(bool choice) {

// Registers button push and returns selection

button\_pushed = true;

selection = choice;

return choice;

}

void Game\_over\_window::wait\_for\_button() { // Waits for button push

show();

while (!button\_pushed) {

Fl::wait();

}

}

//

// Game.h

// Team 4 - 'Please Compile'

//

// Created by SeungJin Kim on 11/10/17.

// Copyright © 2017 SeungJin Kim. All rights reserved.

#ifndef GAME

#define GAME

#include "User\_score.h"

class Game {

int difficulty;

int moves\_left;

string initials;

int move\_count;

int tiles\_incorrect\_position;

Vector<User\_score> top\_scores;

Vector<User\_score> different\_difficulties;

void load\_top\_scores();

public:

Game(int difficulty, string initials);

void set\_moves\_left(int n);

int get\_moves\_left();

void set\_move\_count(int n);

int get\_move\_count();

void set\_num\_tiles\_incorrect\_position(int n);

int get\_num\_tiles\_incorrect\_position();

int get\_difficulty();

Vector<User\_score>& get\_top\_scores();

Vector<User\_score>& get\_different\_difficulties();

};

#endif

//

// Game.cpp

// Team 4 - 'Please Compile'

//

// Created by SeungJin Kim on 11/10/17.

// Copyright © 2017 SeungJin Kim. All rights reserved.

#include "Game.h"

Game::Game(int difficulty, string initials)

: difficulty{ difficulty }, initials{ initials }, move\_count{ 0 }

{

load\_top\_scores(); // Load scores from 'Scores.txt'

tiles\_incorrect\_position = 0;

switch (difficulty) { // Determines number of moves given to user

case 1:

moves\_left = 10; break;

case 2:

moves\_left = 20; break;

case 3:

moves\_left = 40; break;

case 4:

moves\_left = 80; break;

}

}

// Getters and Setters

void Game::set\_moves\_left(int num) {

moves\_left = num;

}

int Game::get\_moves\_left() {

return moves\_left;

}

void Game::set\_num\_tiles\_incorrect\_position(int n) {

tiles\_incorrect\_position = n;

}

int Game::get\_num\_tiles\_incorrect\_position() {

return tiles\_incorrect\_position;

}

int Game::get\_difficulty() {

return difficulty;

}

Vector<User\_score>& Game::get\_top\_scores() {

return top\_scores;

}

Vector<User\_score>& Game::get\_different\_difficulties() {

return different\_difficulties;

}

void Game::set\_move\_count(int num) {

move\_count = num;

}

int Game::get\_move\_count() {

return move\_count;

}

// Loads scores from file

void Game::load\_top\_scores() {

string line;

User\_score user(initials, 0, difficulty);

fstream file("Scores.txt", ios::in | ios::out);

// Open file or create it if DNE

if (file) { // If file is open

User\_score temp("", 0, 0);

while (file >> temp) {

if (temp.get\_difficulty() == difficulty)

top\_scores.push\_back(temp);

else

different\_difficulties.push\_back(temp);

}

file << flush;

file.close();

}

top\_scores.push\_back(user);

}

//

// Difficulty\_window.h

// Team 4 - 'Please Compile'

//

// Created by Daniel Hain-Trevino.

// Copyright © 2017 Daniel Hain-Trevino. All rights reserved.

#ifndef DIFFICULTY\_WINDOW

#define DIFFICULTY\_WINDOW

#include "Simple\_window.h"

class Difficulty\_window : Graph\_lib::Window {

Rectangle background;

Text difficulty\_text;

Button easy\_button;

Button normal\_button;

Button hard\_button;

Button expert\_button;

Rectangle easy\_button\_cover;

Rectangle normal\_button\_cover;

Rectangle hard\_button\_cover;

Rectangle expert\_button\_cover;

Text easy\_button\_text;

Text normal\_button\_text;

Text hard\_button\_text;

Text expert\_button\_text;

bool button\_pushed;

int selection;

void stylize\_objects();

void attach\_objects();

public:

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

void wait\_for\_button();

int change\_state(int choice);

int get\_choice() { return selection; }

void quit();

};

#endif

//

// Difficulty\_window.cpp

// Team 4 - 'Please Compile'

//

// Created by Daniel Hain-Trevino.

// Copyright © 2017 Daniel Hain-Trevino. All rights reserved.

#include "Difficulty\_window.h"

#include "Graph.h"

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

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

background{ Point{0, 0}, Point{x\_max(), y\_max()} },

difficulty\_text{ Point{ 25, 40 }, "Select a difficulty:" },

easy\_button{ Point{ x\_max() / 2 - 304, y\_max() / 2 }, 140, 40, "Easy (10 Moves)",

[](Address, Address pw) {reference\_to<Difficulty\_window>(pw).change\_state(1); } },

normal\_button{ Point{ x\_max() / 2 - 148, y\_max() / 2 }, 140, 40, "Normal (20 Moves)",

[](Address, Address pw) {reference\_to<Difficulty\_window>(pw).change\_state(2); } },

hard\_button{ Point{ x\_max() / 2 + 8, y\_max() / 2}, 140, 40, "Hard (40 Moves)",

[](Address, Address pw) {reference\_to<Difficulty\_window>(pw).change\_state(3); } },

expert\_button{ Point{ x\_max() / 2 + 164, y\_max() / 2}, 140, 40, "Expert (80 Moves)",

[](Address, Address pw) {reference\_to<Difficulty\_window>(pw).change\_state(4); } }, // C++11 features (lambda expressions)

easy\_button\_cover{ Point{ x\_max() / 2 - 304, y\_max() / 2 }, Point{ x\_max() / 2 - 304 + 140, y\_max() / 2 + 40} },

normal\_button\_cover{ Point{ x\_max() / 2 - 148, y\_max() / 2 }, Point{ x\_max() / 2 - 148 + 140, y\_max() / 2 + 40} },

hard\_button\_cover{ Point{ x\_max() / 2 + 8, y\_max() / 2 }, Point{ x\_max() / 2 + 8 + 140, y\_max() / 2 + 40} },

expert\_button\_cover{ Point{ x\_max() / 2 + 164, y\_max() / 2 }, Point{ x\_max() / 2 + 164 + 140, y\_max() / 2 + 40} },

easy\_button\_text{ Point{ x\_max() / 2 - 304 + 15, y\_max() / 2 + 25}, "Easy (10 Moves)" },

normal\_button\_text{ Point{ x\_max() / 2 - 148 + 5, y\_max() / 2 + 25}, "Normal (20 Moves)" },

hard\_button\_text{ Point{ x\_max() / 2 + 8 + 15, y\_max() / 2 + 25}, "Hard (40 Moves)" },

expert\_button\_text{ Point{ x\_max() / 2 + 164 + 5, y\_max() / 2 + 25}, "Expert (80 Moves)" },

button\_pushed{ false },

selection{ 0 }

{

stylize\_objects();

attach\_objects();

}

void Difficulty\_window::stylize\_objects() {

// Changes appearance of screen objects

background.set\_fill\_color(Color::white);

difficulty\_text.set\_font\_size(35);

}

void Difficulty\_window::attach\_objects() { // Attaches objects

attach(background);

attach(difficulty\_text);

attach(easy\_button);

attach(normal\_button);

attach(hard\_button);

attach(expert\_button);

attach(easy\_button\_cover);

attach(normal\_button\_cover);

attach(hard\_button\_cover);

attach(expert\_button\_cover);

attach(easy\_button\_text);

attach(normal\_button\_text);

attach(hard\_button\_text);

attach(expert\_button\_text);

}

void Difficulty\_window::quit() { // Exits window

hide();

}

int Difficulty\_window::change\_state(int choice) {

// Detects button press and returns selection

button\_pushed = true;

selection = choice;

return choice;

}

void Difficulty\_window::wait\_for\_button() { // Waits for button press

show();

while (!button\_pushed) {

Fl::wait();

}

}

//

// Main.cpp

// Team 4 - 'Please Compile'

//

// Created by Luke Grammer

// Copyright © 2017 Luke Grammer. All rights reserved.

#include "Window\_manager.h"

int main() {

try {

Window\_manager manager;

manager.start\_game\_sequence();

return gui\_main();

}

catch (exception& e) { // Catching natively defined errors

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

return 1;

}

catch (...) { // Catching misc. errors

cerr << "An error has occured and caused the program to close unexpectedly. \n";

return 2;

}

}

PART XI – Bibliography