**Project 1 & 2**

**<Labyrinth>**

**CSC-7 43732**

**Timothy Hanmann**

**Date: 06/09/2014**

**Introduction**

Title: Labyrinth

I didn’t do the first project so I ended up putting it together for one big project.

This is a challenge/puzzle game.

The main goal is to navigate through the maps having various obstacles and traps and getting the end of the maze.

The user has a certain amount of health points (HP) which if it reaches 0, the user loses and is therefore Game Over. The time it takes for you to reach the end will also be logged for you own benefit.

This game is meant to be a fun experience

**Summary**

Project Size: About 400+

The number of variables: 15

The number of method: 7

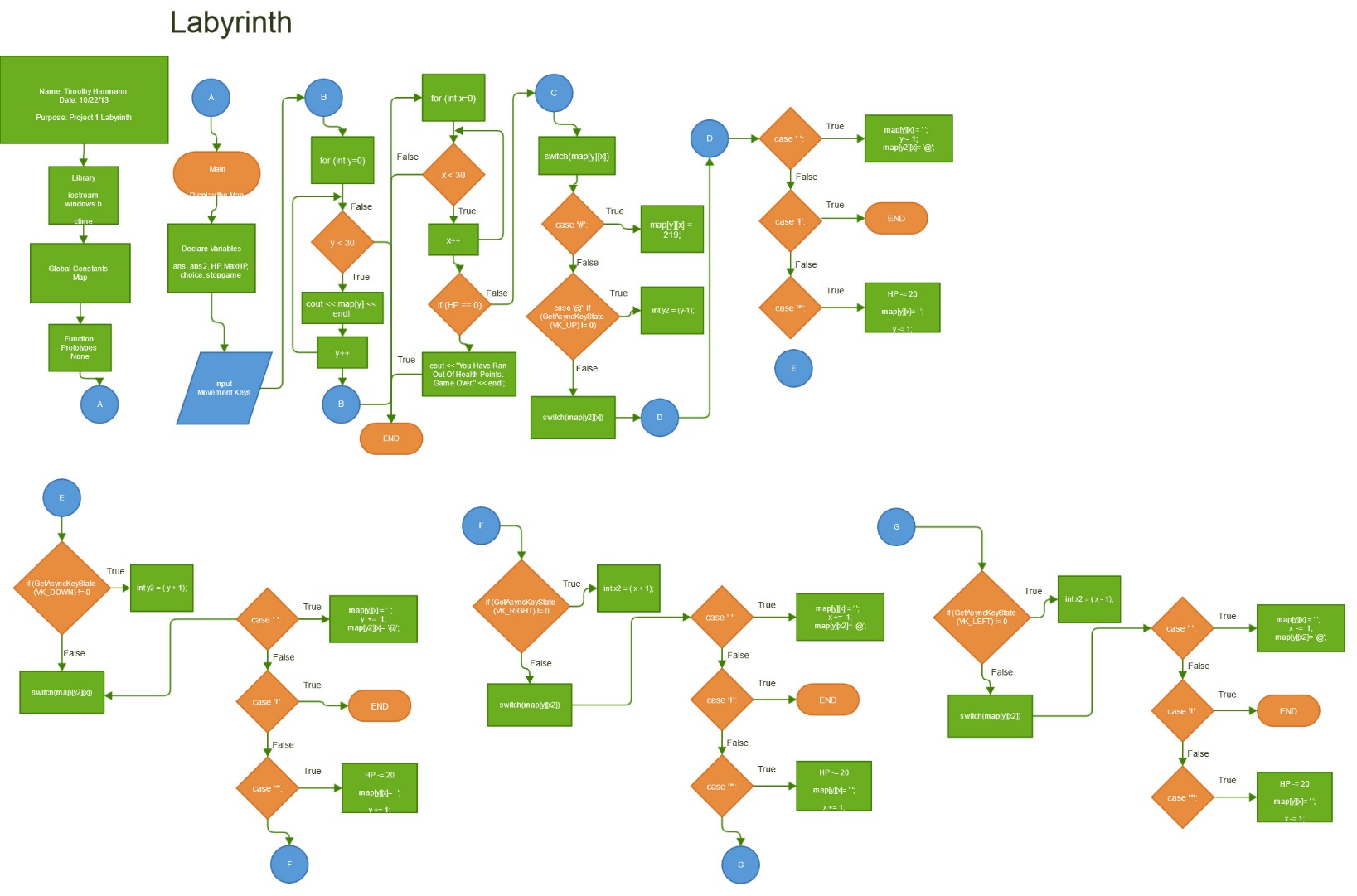
The project gave me a good chance to develop my skill further and really push myself to try something different and difficult. If I had time I would’ve finished it but I kept finding myself adding more and more to it since there was so many possibilities. I also did a lot of organizing to make it look neater and shorten my code a bit by using multiple files, I was going to add a login by trying to use hash salts but I couldn’t understand. Overall I learned a lot from this project as well as this course and will only grow from it.

The length of this project took entirely about a week in a half, it would take longer if I had more time but I had to stop adding things at some point to have some kind of end to my project. It took me awhile to learn certain algorithms like the Depth-First Search to use as a procedural generator for my random maze, I couldn’t get that implemented unfortunately but I still intend to do in it in the future.

**Description**

The main reason I chose this game was to improve on what was done already and to elaborate on it and expand what my program can do.

**Flow Chart**



**Pseudo Code**

*Initialization*

*If user presses Left Arrow*

*@ model moves left*

*Else if user press Right Arrow*

*@ model moves right*

*Else if user press Up Arrow*

*@ model moves up*

*Else if user press Down Arrow*

*@ model moves down*

*If player touches \**

*Reduces HP by 20*

*Else if player reaches !*

*Game Over prompt comes up*

*Else*

*Wrong choice of input*

**Major Variables**

|  |  |  |  |
| --- | --- | --- | --- |
| **Type** | **Variable Name** | **Description** | **Location** |
| integer | Level | The amount of levels | Level = 1 |
|  | HP | The current amount of health points | HP = 100 |
|  | MaxHP | The maximum amount of health points | MaxHP = 100 |
|  | Gamespeed | The speed of the game | Gamespeed = 100 |
| bool | stopgame | Expression for the game to end | stopgame = false; |
|  | flag | Timer | while (ans == 'N' || ans == 'n'); |
| char | ans | Player choice to play game or not | flag = false; |
|  | map | Displays the outlook of the map | char map[30][30] |
| float | Secondspassed | The amount of time | secondsPassed = (clock() - startTime) / CLOCKS\_PER\_SEC; |
|  | secondstodelay | Amount of delay from game | if(secondsPassed >= secondsToDelay){ |

**C++ Constructs**

|  |  |  |
| --- | --- | --- |
| **Chapter** | **Syntax and keywords** | **Location** |
| 1 | Arithmetic Operators (-, +,, \*, /) | y2 = (y-1); |
|  | If/else statements | if (HP == 0) |
|  | Equality Operators (==,!=, <, <=, >, >=) | (int y = 0; y < 30; y++) |
|  | Int primitive type | int Level |
|  | Assignment Operators (+=) | y += 1; |
|  | While loop | while(flag) |
| 2 | Increment Operators (++) | x++ |
|  | Switch | switch(map[y][x]) |
|  | Break | map[y2][x]= '@';  }break; |
| 3 | Block | 1secondsPassed = (clock() - startTime) / CLOCKS\_PER\_SEC;  if(secondsPassed >= secondsToDelay){ |
|  | For statement | for (int x = 0; x < 30; x++) |
| 4 | Using Namespace | using namespace std; |

**Reference**

1. Textbook

2. Internet/Web

3. Materials

**Program**

//Labyrinth

//Libraries

#include "scoreboard.h"

#include "levelmenu.h"

//Function from Joshua Camacho don't credit for line count

#ifdef \_WIN32

#endif

#ifdef \_WIN32

static HANDLE hConsoleOutput = 0;

static CONSOLE\_SCREEN\_BUFFER\_INFO csbi;

#endif

using namespace std;

//Global Constants

//Execution starts here

int main(int argc, char\*\* argv) {

//Declare variables

string playerName;

int sec, min, mChoice;

bool gameOver = false;

int HP=100, MaxHP=100, gamespeed=100;

char ans,uChoice;

time\_t start = 0;

time\_t end = 0;

time\_t elapsed = 0;

vector<int> numPlayers;

string namePlayers;

//Opens up file to save/load Times

ofstream fout;

fout.open("SavedPlayers.txt", ios::app);

ifstream infile;

infile.open ("SavedPlayers.txt");

//fillBoard(numPlayers, namePlayers);

//printBoard(numPlayers, namePlayers);

do{

//Prompt for new player

cout << "Are you a new player to the game? Y or N" << endl;

cin >> uChoice;

//If new player, adds you too save file

if(uChoice == 'Y' || uChoice == 'y'){

// storage();

cout << "What is your name?" << endl;

cin >> playerName;

fout << playerName << endl;

LevelMenu();

cout << "The controls for the game are the arrow keys." << endl; //Controls

cin >> mChoice;

}else

cout << "What is your saved name?" << endl;

LevelMenu();

cin >> mChoice;

//Arrow keys for Map 1

while (gameOver == false && mChoice == 1){

//clear();

start = time(NULL);

for( int y=0;y<30;y++){

cout << map1[y] << endl;

}

cout << "Time: " << time << endl;

cout << "HP: " << HP << "/" << MaxHP << endl;

for(int y = 0;y < 30;y++){

for(int x = 0; x < 30;x++){

//Walls

switch(map1[y][x]){

case '#':{

map1[y][x] = 219;

}

case '@': //The Player

{ //Arrow Mapping for Up Key

if (GetAsyncKeyState (VK\_UP) != 0){

int y2 = (y-1);

switch (map1[y2][x]){

case ' ':

{

map1[y][x] = ' ';//old space

y -= 1;

map1[y2][x]= '@';//Occupies new space

}break;

case '!':{//When the end is reached

gameOver=true;

}break;

case '\*':{//Loss of HP

HP -= 20;

map1[y][x] = ' ';

y -= 1;

map1[y2][x]= '@';

}break;

}

}//Arrow Mapping for Down Key

if(GetAsyncKeyState (VK\_DOWN) != 0){

int y2 = (y + 1);

switch(map1[y2][x]){

case ' ':{

map1[y][x] = ' ';

y += 1;

map1[y2][x]= '@';

}break;

case '!':{

gameOver=true;

}break;

case '\*':{//Loss of HP

HP -= 20;

map1[y][x] = ' ';

y += 1;

map1[y2][x]= '@';

}break;

}

} //Arrow Mapping for Right Key

if (GetAsyncKeyState (VK\_RIGHT) != 0){

int x2 = (x + 1);

switch(map1[y][x2]){

case ' ':{

map1[y][x] = ' ';

x += 1;

map1[y][x2]= '@';

}break;

case '!':{

gameOver=true;

}break;

case '\*':{//Loss of HP

HP -= 20;

map1[y][x] = ' ';

x += 1;

map1[y][x2]= '@';

}break;

}

} //Arrow Mapping for Left Key

if (GetAsyncKeyState (VK\_LEFT) !=0){

int x2 = (x - 1);

switch (map1[y][x2]){

case ' ':{

map1[y][x] = ' ';

x -= 1;

map1[y][x2]= '@';

}break;

case '!':{

gameOver=true;

}break;

case '\*':{//Loss of HP

HP -= 20;

map1[y][x] = ' ';

x -= 1;

map1[y][x2]= '@';

}break;

}

}

}break;

}

}clear();

}

Sleep(gamespeed);

end = time(NULL);

elapsed = end - start;

}

//cout << "This is how long it took you: " << elapsed << " seconds" << endl << endl;

//Arrow Keys for Map 2

/\*while (gameOver == false && mChoice == 2){

for( int y=0;y<30;y++){

cout << map2[y] << endl;

}

for(int y = 0;y < 30;y++){

for(int x = 0; x < 30;x++){

switch(map2[y][x]){

case '#':{

map2[y][x] = 219;

}

case '@':

{ //Arrow Mapping for Up Key

if (GetAsyncKeyState (VK\_UP) != 0){

int y2 = (y-1);

switch (map2[y2][x]){

case ' ':

{

map2[y][x] = ' ';

y -= 1;

map2[y2][x]= '@';

}break;

case '!':{

gameOver=true;

}break;

case '\*':{//Loss of HP

HP -= 20;

map2[y][x] = ' ';

y -= 1;

map2[y2][x]= '@';

}break;

}

}//Arrow Mapping for Down Key

if(GetAsyncKeyState (VK\_DOWN) != 0){

int y2 = (y + 1);

switch(map2[y2][x]){

case ' ':{

map2[y][x] = ' ';

y += 1;

map2[y2][x]= '@';

}break;

case '!':{

gameOver=true;

}break;

case '\*':{//Loss of HP

HP -= 20;

map2[y][x] = ' ';

y += 1;

map2[y2][x]= '@';

}break;

}

} //Arrow Mapping for Right Key

if (GetAsyncKeyState (VK\_RIGHT) != 0){

int x2 = (x + 1);

switch(map2[y][x2]){

case ' ':{

map2[y][x] = ' ';

x += 1;

map2[y][x2]= '@';

}break;

case '!':{

gameOver=true;

}break;

case '\*':{//Loss of HP

HP -= 20;

map2[y][x] = ' ';

x += 1;

map2[y][x2]= '@';

}break;

}

} //Arrow Mapping for Left Key

if (GetAsyncKeyState (VK\_LEFT) !=0){

int x2 = (x - 1);

switch (map2[y][x2]){

case ' ':{

map2[y][x] = ' ';

x -= 1;

map2[y][x2]= '@';

}break;

case '!':{

gameOver=true;

}break;

case '\*':{//Loss of HP

HP -= 20;

map2[y][x] = ' ';

x -= 1;

map2[y][x2]= '@';

}break;

}

}

}break;

} \*

}

}

end = time(NULL);

elapsed = end - start;

Sleep(gamespeed);

clear();

cout << "This is how long it took you: " << elapsed << " seconds" << endl << endl;

}\*///cout << "Would you like to play the game again?" << endl;

// cin >> ans;

}while (ans == 'Y' || ans == 'y');

return 0;

}

//Function Definitions

/\*void storage(){

}\*/

/\*void fillBoard(vector<int>&numPlayers,string &playerName){

char uChoice;

storage();

cout << "Are you are new player to the game? Y or N" << endl;

cin >> uChoice;

switch(uChoice){

case 1: if (uChoice == 'Y' || uChoice == 'y'){

cout << "What is your name?" << endl;

}

}

//Players name

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

cout << endl << "Enter the names of players: ";

cin >> playerName;

}

}\*/

void printBoard(vector<int>&numPlayers,string playerName){

cout << "Scoreboard: " << endl;

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

for(int y = 0;y < numPlayers.size();y++){

cout << numPlayers[y] << " " << playerName[i] << endl;

}

}

}

void LevelMenu(){

int mChoice, test;

char choice;

// cout << test << endl;

cout << "Would you like to choose this map?"<< endl;

cout << "Map 1: " << endl << endl

<< "############################\n"

<< "#@ \* # #\n"

<< "# # ###### #### ####### #\n"

<< "# #\* # # ## #\n"

<< "# ### # ######\* #\* # #\n"

<< "# \*# ######## # # #\n"

<< "# ## # # # # # #\n"

<< "# #\* \*### # # # # #\n"

<< "# #########\* # #### # #\n"

<< "# # #\* \*# #\n"

<< "############## \*########## #\n"

<< "### # # # # # ## #\n"

<< "## ### ### ## # # # #\n"

<< "#### # ## # !#\n"

<< "############################\n" << endl;

cout << "Press 2 to view the other map or press 1 to keep this map." << endl;

// cout << "Press Y "

cin >> mChoice;

switch(mChoice){

case 1:

//showcases the map to the user

cout << "Map 1: " << endl << endl

<< "############################\n"

<< "#@ \* # #\n"

<< "# # ###### #### ####### #\n"

<< "# #\* # # ## #\n"

<< "# ### # ######\* #\* # #\n"

<< "# \*# ######## # # #\n"

<< "# ## # # # # # #\n"

<< "# #\* \*### # # # # #\n"

<< "# #########\* # #### # #\n"

<< "# # #\* \*# #\n"

<< "############## \*########## #\n"

<< "### # # # # # ## #\n"

<< "## ### ### ## # # # #\n"

<< "#### # ## # !#\n"

<< "############################\n" << endl;

//asks if user wants to pick map or choose a new one

cout << "\nDo you choose this map? Type Y.";

cin >> choice;

break;

case 2:

cout << "Map 2: " << endl << endl

<< "############################\n"

<< "#! ## # ##############\n"

<< "## # # #\n"

<< "### ###### ######## # #\n"

<< "##### ####### # # #\n"

<< "# ### ## # # # #\n"

<< "## ## ######### # # #\n"

<< "## ## ### #\n"

<< "### ######## # #\n"

<< "### ######## ##########\n"

<< "### #### ### ## #\n"

<< "##### # ########## ### #\n"

<< "######## ################# #\n"

<< "#@ #\n"

<< "############################\n" << endl;

cout << "\nDo you choose this map? Type Y.";

cin >> choice;

break;

default: cout << "That is not a map choice." << endl;

break;

}

}

float timer(float sec,float min, bool sub){

sub == false;

for(sec = 0; sec >= 60;sec++){

if (sec == 60 && sub == false){

min += 1;

sec = 0;

}

cout << min << " ; " << sec << endl;

}

}

//Function from Joshua Camacho don't credit for line count

void clear(){

#ifdef \_WIN32

if ( hConsoleOutput == 0 )

{

hConsoleOutput = GetStdHandle(STD\_OUTPUT\_HANDLE);

GetConsoleScreenBufferInfo(hConsoleOutput, &csbi);

}

#endif

fflush(stdout);

#ifdef \_WIN32

{

COORD point;

DWORD dummy;

DWORD count;

point.X = 0;

point.Y = 0;

count = csbi.dwSize.X \* csbi.dwSize.Y;

FillConsoleOutputCharacter(hConsoleOutput, ' ', count, point, &dummy);

SetConsoleCursorPosition(hConsoleOutput, point);

}

#else

printf("\033[H\033[2J");

#endif

fflush(stdout);

}

/\*if (choice == 'Y' || choice == 'y'){

LevelMenu();

cout << "The controls for the game are the arrow keys." << endl; //Controls

cin >> mChoice;

start = time(NULL);

} else{

cout << "The controls for the game are the arrow keys." << endl; //Controls

LevelMenu();

cin >> mChoice;

start = time(NULL);

} \*/

//LevelMenu.h

/\*

\* levelmenu.h prototypes

\*/

#pragma once

#include "scoreboard.h"

#include<iostream>

#include<fstream>

using namespace std;

//Function Prototypes

unsigned char map1[30][30]=

{"############################",

"#@ # \* # #",

"# # ###### #### ####### #",

"# #\* # # ## #",

"# ### # ######\* #\* # #",

"# \*# # ######## # # #",

"# ## # # # # # #",

"# #\* \*### # # # # #",

"# #########\* # #### # #",

"# # #\* \*# #",

"### ##### #### \*########## #",

"### # # # # # ## #",

"## ### ### ## # # # #",

"#### # ## # !#",

"############################"};

unsigned char map2[30][30]=

{"############################",

"#! ## # ##############",

"## # # #",

"# # ###### ######## # #",

"# ### ####### # # #",

"# ## ## # # # #",

"## ## ######### # # #",

"## ## ### #",

"### ######## # #",

"### ######## ##########",

"### #### ### ## #",

"##### # ########## ### #",

"######## ################# #",

"#@ #",

"############################",};

void LevelMenu();

void clear();