**Project 1**

**<Labyrinth>**

**CSC-5 47981**

**Timothy Hanmann**

**Date: 10/13/13**

**Introduction**

Title: Labyrinth

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 and one that tests people’s wits in a simple puzzle game.

**Summary**

Project Size: About 200+

The number of variables: 9

The number of method: 4

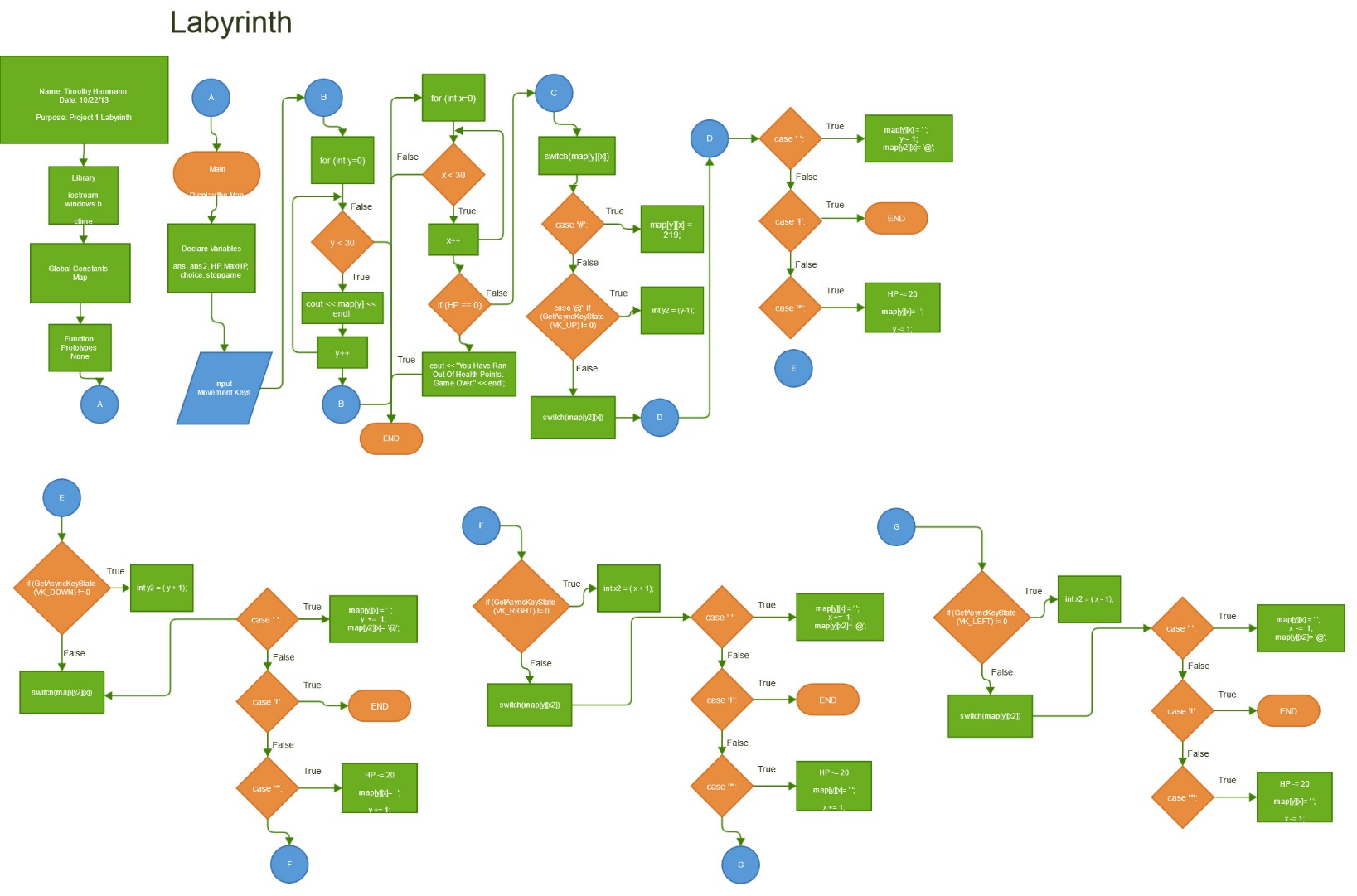
The project gave me a good chance to develop my skill further and really push myself to try something different and difficult. I’m hoping that I can better it in the future when I have learned more concepts and methods to efficiently create this game. Kept running into many different problems with the program bugging.

The length of this project took entirely about a week in a half with the minor obstacles as well as the challenge of learning different types of parts that I didn’t understand to well. I learned a lot from this experience and hope to improve from it so I can create something impressive. I learned a few new things while working on this project and at the same time used ones we’ve learned already and just have learned other ways to use them.

**Description**

The main reason I chose this game was to try to use movement keys and programmed it the character to move in whichever direction was pressed.

**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 <iostream>

#include <fstream>

#include <string>

#include <ctime>

#include <windows.h>

using namespace std;

//No Global Constants

char map[30][30]= {"#############################",

"# # \* # #",

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

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

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

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

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

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

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

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

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

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

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

"# ## ##",

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

//No Function Prototypes

//Program starts here

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

//Declare variables

char ans;

bool stopgame = false;

bool flag = true;

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

int Level = 1;

float secondsPassed;

float secondsToDelay = atof(argv[1]);

//Timer Start

clock\_t startTime = clock();

//Prompt if user wants to play

do{

cout << "Welcome to Labyrinth! In this game you will be maneuvering through the maze"

<< " to escape and reach the end." << endl;

cout << "Are you ready to play the game?" << endl;

cin >> ans;

}while (ans == 'N' || ans == 'n');

//Game begins here

while (stopgame == false && Level == 1){

//Countdown for Timer

cout << "Time to delay: " << secondsToDelay << endl;

while(flag){

secondsPassed = (clock() - startTime) / CLOCKS\_PER\_SEC;

if(secondsPassed >= secondsToDelay){

cout << secondsPassed << " seconds have passed" << endl;

flag = false;

}

}

system("cls");

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

{

cout << map[y] << endl;

}

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

//If HP reaches 0 = Game Over.

if (HP == 0){

system("pause");

cout << "You Have Ran Out Of Health Points. Game Over." << endl;

return EXIT\_SUCCESS;

}

//Mapping for @ to keep within walls

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

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

switch(map[y][x]){

case '#':{ // The Look of the Walls

map[y][x] = 219;

}

case '@':

{ //Arrow Mapping for Up Key

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

{

int y2 = (y-1);

switch (map[y2][x]){

case ' ':

{

map[y][x] = ' ';

y -= 1;

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

}break;

case '!':{

Level = 2;

cout << secondsPassed << " seconds have passed" << endl

<< endl;

}break;

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

HP -= 20;

map[y][x] = ' ';

y -= 1;

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

}break;

}

}//Arrow Mapping for Down Key

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

int y2 = (y + 1);

switch(map[y2][x]){

case ' ':{

map[y][x] = ' ';

y += 1;

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

}break;

case '!':{

Level = 2;

cout << secondsPassed << " seconds have passed" << endl

<< endl;

}break;

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

HP -= 20;

map[y][x] = ' ';

y += 1;

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

}break;

}

} //Arrow Mapping for Right Key

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

int x2 = (x + 1);

switch(map[y][x2]){

case ' ':{

map[y][x] = ' ';

x += 1;

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

}break;

case '!':{

Level = 2;

cout << secondsPassed << " econds have passed" << endl

<< endl;

}break;

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

HP -= 20;

map[y][x] = ' ';

x += 1;

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

}break;

}

} //Arrow Mapping for Left Key

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

int x2 = (x - 1);

switch (map[y][x2]){

case ' ':{

map[y][x] = ' ';

x -= 1;

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

}break;

case '!':{

Level = 2;

cout << secondsPassed << " seconds have passed" << endl

<< endl;

}break;

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

HP -= 20;

map[y][x] = ' ';

x -= 1;

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

}break;

}

}

}break;

}

}

}

Sleep(gamespeed);

}