Project 2

<GMO Dungeon>

CSC-5, 48101

Thomas, Jeffrey

12/12/16

Table of Contents

Introduction…………………………………………………………….3

How Big is it, Jeff?…………………………………………………….3

Concepts Utilized……………………………………………………...4

More Concepts……………………………………………………...…5

Psuedocode………………………………………….…………....6 – 9

Full Code………………………………………………………...10 - 30

Acknowledgements………………………………………………….31

Introduction

Title: GMO Dungeon

GMO Dungeon is a dungeon crawler style text RPG. The player takes on the role of a beta tester for an unusual service in which you can genetically engineer custom girlfriends on command. Trouble arises in the lab and you must escape the research facility with the aid of your new girlfriend.

There are five unique floors of increasing difficulty, each with their own totem that must be defeated in order to continue to the next floor. As you defeat enemies and explore the dungeon, your girlfriend levels up and increases in power.

Primary Features are:

-Free roaming dungeon with interactive map

-Unique scenarios

-Turn based combat

-Custom party members

-Progressive difficulty

-Save game/Load game

How big is it, Jeff?

Project Size: Approx. 1529 lines w/o end-to-end

Approx. 1200 lines w/o text,comments,formatting

No. Of Variables: 24

No. Of Functions: 14

Concepts Utilized

Concept Line Number

|  |  |
| --- | --- |
| cin | 182 (and throughout) |
| cout | 179 (and throughout) |
| cin.get() | 216, 225, 290, 301 |
| cin.ignore() | 183, 243, |
|  |  |
| srand() | 78 |
| rand() | 613 (and throughout) |
|  |  |
| time | 78 |
|  |  |
| getline() | 871 |
|  |  |
| Ifstream | 868, 1519 |
| Ofstream | 1484 |
| .open() | 869, 891, 1485, 1520 |
| .close() | 876, 899, 1505, 1539 |
|  |  |
| Int | 25 (and throughout) |
| Char | 40 (and throughout) |
| Bool | 84 (and throughout) |
| String | 33 (and throughout) |
|  |  |
| If | 203 (and throughout) |
| If/else | 1102 (and throughout) |
| If/else if | 917 (and throughout) |
| Switch | 184 (and throughout) |
| Ternary | 942, 960 |
|  |  |
| For | 311 (and throughout) |
| While | 994 |
| Do-while | 181 (and throughout) |

More Concepts

Concept Line Number

|  |  |
| --- | --- |
| #include | 9 – 13 |
| Const | 19, 20 |
| Arithmetic operators | 471 (and throughout) |
| Relational operators | 203 (and throughout) |
| Logical operators | 201 (and throughout) |
| Input validation | 271, 443, 761 |
| Increment/Decrement | 311 (and throughout) |
| Nesting | 181 (and throughout) |
| Arrays (up to 3 dimensions) | 71 (and throughout) |

Pseudocode

Initialize

Declare Arrays

Initialize Boss Arrays

Declare Variables

Output ASCII Game Title

Prompt user

Take input

Choice 1:

Set new game flags

Choice 2:

Load game

Choice 3:

Notify input is invalid

While choice is not valid

Intro Story

Generate companion

Output Stats

“Keep it?”

Yes: Add to party

No: Delete companion, generate another, “Keep it?”

Default: Notify input is invalid

While choice is not valid

More Plot

Repeat for each floor

Output current floor

Initialize map

Initialize bool arrays

Find boss and assign coordinates

Spawn player

Display Map

Reset bool array

Repeat until player wants to stop

Prompt input

Choice 1: Save Game

Choice 2: Heal

Choice 3: Leave

Choice 4: End Program

Default: Notify input is invalid

Repeat until boss is defeated

Output controls

Input movement

Choice 1:

Move player marker +1 y

Check for walls or edges

Output map

Choice 2:

Move player marker -1 y

Check for walls or edges

Output map

Choice 3:

Move player marker -1 x

Check for walls or edges

Output map

Choice 4:

Move player marker +1 x

Check for walls or edges

Output map

Default:

Notify choice is invalid

Check which tile player is on

Option 1:

Initiate Battle

Return outcome

Outcome 1: Load Game

Outcome 2: Quit Game

Outcome 3: Update XP

Option 2:

Outcome 1: Nothing Happens

Outcome 2: Restore health

Outcome 3: Level up

Option 3:

Repeat until player wants to stop

Prompt input

Choice 1: Save Game

Choice 2: Heal

Choice 3: Leave

Choice 4: End Program

Default: Notify input is invalid

Option 4:

Boss Battle

Return outcome

Outcome 1: Load Game

Outcome 2: Quit Game

Outcome 3: Update XP

Exit Program

Full Code (Minus Function Definitions)

/\*

\* File: main.cpp

\* Author: Jeffrey Thomas

\* Created on December 3, 2016, 12:36 AM

\* Purpose: Project 2

\*/

//System Libraries

#include <iostream> //Input/Output objects

#include <fstream> //File I/O

#include <string> //String Manipulation

#include <ctime> //Random Seed

#include <cstdlib> //Random Generator

using namespace std; //Namespace used in system library

//Global constants

int const LENGTH=5; //Length of 3D array representing dungeon

int const WIDTH=5; //Width of 3D array representing dungeon

//Structures

//Monster holds all values of the GF's you collect

struct Monster{

int hpt; //Max Health Points

int chp; //Current health

int atk; //Base Attack

int def; //Base Defense

int spd; //Base Speed

int mxp; //Max EXP

int cxp; //Current EXP

int lvl; //Current Level

string name; //Monster Name

string flvr; //Monster Flavor Text

};

//Function prototypes

//Execution begins here

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

{

//Initialize Random Seed

srand(time(0));

//Declare Arrays

char roomTp[5][LENGTH][WIDTH];

//Location checks

bool visitd[LENGTH][WIDTH]={};

bool inside[LENGTH][WIDTH]={};

bool used[LENGTH][WIDTH]={};

//Boss Monsters

Monster boss[5];

//Initialize Bosses

//Initialize LILIM

boss[0].hpt=20;

boss[0].chp=boss[0].hpt;

boss[0].atk=8;

boss[0].def=12;

boss[0].spd=12;

boss[0].name="L.I.L.I.M.";

boss[0].flvr="A wretched mutant GMO with small wings. She is the leader of "

"the lesser monsters of Floor B1";

//Initialize ISHTAR

boss[1].hpt=40;

boss[1].chp=boss[1].hpt;

boss[1].atk=20;

boss[1].def=12;

boss[1].spd=18;

boss[1].name="I.S.H.T.A.R.";

boss[1].flvr="A reptilian GMO whose fangs can kill instantly. She is the "

"leader of the lesser monsters of Floor B2";

//Initialize FREYA

boss[2].hpt=60;

boss[2].chp=boss[2].hpt;

boss[2].atk=30;

boss[2].def=30;

boss[2].spd=30;

boss[2].name="F.R.E.Y.A.";

boss[2].flvr="A massive GMO with skin made of steel. She is the leader of "

"the lesser monsters of Floor B3";

//Initialize EROS

boss[3].hpt=80;

boss[3].chp=boss[3].hpt;

boss[3].atk=42;

boss[3].def=35;

boss[3].spd=65;

boss[3].name="E.R.O.S.";

boss[3].flvr="A vile GMO with large pores endlessly leaking. She is the "

"leader of the lesser monsters of Floor B4";

//Initialize APHRODITE

boss[4].hpt=100;

boss[4].chp=boss[4].hpt;

boss[4].atk=65;

boss[4].def=50;

boss[4].spd=80;

boss[4].name="A.P.H.R.O.D.I.T.E.";

boss[4].flvr="Queen of the mutant nest that's formed in the depths of the "

"abandoned research facility. Because she's attached to the walls, "

"she sends scavengers to collect food.";

//GF Name Prefixes

string syllabl[18]={"LAC","KAM","FAN","KAR","PIP","LOR","DOR","ANN","TOR",

"MAR","DAN","LEX","RUB","LIL","KIK","VIK","HID","ARR"};

//GF "version numbers" (for naming)

char numCh[]={'0','1','2','3','4','5','6','7','8','9'};

//Enemy Names

string enNames[5] ={"Mutant GF","Wandering Security Drone","Giant Rat",

"Daughter of Aphrodite","GF Ghost"};

//Declare Variables

char movDir; //Direction player chooses to move

char choice; //Choice input for various prompts

int outcome; //Determines battle outcome branch

int floor; //The current floor of the dungeon

int room=5; //Number of floors in the dungeon

int bosPosX,bosPosY; //Boss location on grid

int curPosX,curPosY; //Player's location on grid

bool intro; //Holds whether player has completed intro or not

bool isLoad; //Holds whether player has already loaded gam

string maps="maps.dat"; //Holds file name for map data

string data="data.dat"; //Holds file name for save data

string titl="title.dat"; //Holds file name for game title ASCII art

Monster gf1; //Player's GF

Monster en; //Lesser enemies

//Output Title

dispTtl(titl);

//Prompt user to select an option

cout<<"1. New Game"<<endl;

cout<<"2. Load Game"<<endl;

do{

cin>>choice; //Select an option

cin.ignore(); //Allows cin.get() to work correctly

switch(choice){

case 49: //49 is ASCII value of '1'

floor=0; //Set floor to 0

intro=1; //Indicate introduction has not been completed

isLoad=1; //Indicate save file was not loaded

break;

case 50:

cout<<endl;

//Load File

load(visitd,intro,gf1,floor,data);

isLoad=0; //Indicate save file was loaded

break;

default:

//Output invalid input statement

cout<<"INVALID INPUT"<<endl<<endl;

break;

}

}while(choice>50||choice<49); //Check to make sure input is valid

if(intro==1){ //Check if introduction has already been done

//Introduction

cout<<endl;

cout<<"Hello! Welcome to GMO (Girlfriends Made to Order)!"<<endl<<endl;

cout<<"I'm the lead researcher here. I'd like to thank you for agreeing "

<<endl<<"to participate in our beta testing. We take it upon ourselves"

<<endl<<"to provide a much needed service for people like you and I. "

<<endl<<"We're great guys who just can't seem to get a gf no matter how"

<<endl<<"hard we try! Let's take a quick tour of the facility and then"

<<endl<<"we can let you start the testing process!";

cout<<endl<<endl;

cout<<"Press ENTER to continue";

cin.get(); //Wait for user to press enter

cout<<endl;

cout<<"(You're led to a room with a huge glass chamber inside)"<<endl<<endl;

cout<<"This is our GMOGen Girlfriend Generator, or GMOGGG. This is what we"

<<endl<<"will utilize for this beta test. Go ahead and press that big"

<<endl<<"button in the middle of the console.";

cout<<endl<<endl;

cout<<"Press ENTER to start the beta test";

cin.get(); //Wait for user to press enter

cout<<endl;

cout<<"(The machine emits a puff of smoke and a figure appears)"<<endl;

//Randomly generate party member

gnratGf(gf1,syllabl,numCh);

//Display generated party member

cout<<"You have created: "<<gf1.name<<endl; //Name

cout<<"Vitality : "<<gf1.hpt<<endl; //Max Health

cout<<"Strength : "<<gf1.atk<<endl; //Attack

cout<<"Toughness: "<<gf1.def<<endl; //Defense

cout<<"Agility : "<<gf1.spd<<endl; //Speed

//Prompt user for response

cout<<"Will you keep this one? ('y' to keep or 'n' to try again)";

cout<<endl;

do{

cin>>choice; //Input choice

cin.ignore(); //Allows cin.get() to work correctly

cout<<endl;

switch(choice){

case 'y': //If Yes

//Notify that party member has been added

cout<<gf1.name<<" has been added to your party!";

cout<<endl<<endl;

break;

case 'n': //If no

//Notify that another GF will be created

cout<<gf1.name<<" is sent into a room labeled 'INCINERATOR'";

cout<<endl<<endl;

//Randomly generate another GF

cout<<"(The machine emits a puff of smoke and a figure appears)"<<endl;

gnratGf(gf1,syllabl,numCh);

cout<<"You have created: "<<gf1.name<<endl;

cout<<"Vitality : "<<gf1.hpt<<endl;

cout<<"Strength : "<<gf1.atk<<endl;

cout<<"Toughness: "<<gf1.def<<endl;

cout<<"Agility : "<<gf1.spd<<endl;

cout<<"Will you keep this one? ('y' to keep or 'n' to try again)";

cout<<endl;

break;

default: //If input is invalid

cout<<"INVALID INPUT";

cout<<endl<<endl;

}

}while(choice!='y'); //Reiterate until player is satisfied

//More plot

cout<<"(The lights shut off and loud sirens begin to blare)";

cout<<endl<<endl;

cout<<"RED ALERT. RED ALERT. APHRODITE HAS AWOKEN. SHUT DOWN COMMENCING.";

cout<<endl<<endl;

cout<<"(You're confused by the sudden alert but you know that whatever it"

<<endl<<"is can't possibly be good. All of the exits are closing one"

<<endl<<"by one. Your tour guide has abandoned you. You must find a "

<<endl<<"way out before it's too late. You see an elevator that hasn't"

<<endl<<"been shut yet.)";

cout<<endl<<endl;

cout<<"Press ENTER to run";

cin.get(); //Wait for user to press enter

cout<<endl;

cout<<"(You make it into the elevator right before it closes. Your GMO"

<<endl<<"seems to have followed you as well. You look at the elevator"

<<endl<<"panel and notice it only has one button. You have no idea"

<<endl<<"where it will go but you seem to have no choice now.)";

cout<<endl<<endl;

cout<<"Press ENTER to press the elevator button";

cin.get(); //Wait for user to press enter

cout<<endl;

cout<<"(You arrive to a dark corridor. The sign on the wall says \"B1\")";

cout<<endl<<endl;

intro=0; //Indicate that introduction has been completed

}

//Game Start

for(floor;floor<room;++floor){ //Loop repeats for number of floors

//Output Current Floor

cout<<"Floor: B"<<floor+1<<endl;

//If new game, load default data

if(isLoad==1){

//Fill in map data from file

fillFloor(roomTp,room,maps);

//Update ifVisit

ifVisit(roomTp,visitd,floor);

//Update ifInsid

ifInsid(roomTp,inside,floor);

//Find boss tile and assign coordinates

findB(roomTp,floor,bosPosX,bosPosY);

//Spawn player at Hub

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

for(int k=0;k<WIDTH;k++){

if(roomTp[floor][j][k]=='H'){

inside[j][k]=0;

visitd[j][k]=0;

}

}

}

//Update floor map with correct symbols

updtFloor(roomTp,visitd,inside,floor);

//Display floor map

dispMap(roomTp,floor);

cout<<endl;

//Initialize/Reset rooms to be unused

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

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

used[i][j]=1;

}

}

}

//If load game, initialize loaded data

if(isLoad==0){

//Fill floor map with file data

fillFloor(roomTp,room,maps);

//Set all visited rooms to 'used'

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

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

used[i][j]=visitd[i][j];

}

}

//Update ifInsid

ifInsid(roomTp,inside,floor);

//Find boss tile and assign coordinates

findB(roomTp,floor,bosPosX,bosPosY);

//Spawn player at Hub

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

for(int k=0;k<WIDTH;k++){

if(roomTp[floor][j][k]=='H'){

inside[j][k]=0;

visitd[j][k]=0;

used[j][k]=1;

}

}

}

//Update Map

updtFloor(roomTp,visitd,inside,floor);

//Display Map

dispMap(roomTp,floor);

cout<<endl;

isLoad=1; //Change flag to allow game to be loaded again

}

//Display Hub Options

do{

cout<<"This room appears to still have working "

<<"equipment inside. What would you like to do?";

cout<<endl;

//Prompt user for input

cout<<"1. Save Game"<<endl;

cout<<"2. Rest"<<endl;

cout<<"3. Leave Hub"<<endl;

cout<<"4. Quit Game"<<endl;

cin>>choice; //Input choice

cout<<endl;

switch(choice){

case 49: //If choice is 1

//Update data on floor map

updtFloor(roomTp,visitd,inside,floor);

//Display floor map

dispMap(roomTp,floor);

cout<<endl;

//Save Game

save(visitd,intro,gf1,floor,data);

//Affirm that game has been successfully saved

cout<<"GAME SAVED";

cout<<endl<<endl;

break;

case 50: //If choice is 2

//Update data on floor map

updtFloor(roomTp,visitd,inside,floor);

//Display floor map

dispMap(roomTp,floor);

cout<<endl;

//Set maximum health to current health

gf1.chp=gf1.hpt;

//Affirm that GF has been healed

cout<<gf1.name<<" has been fully healed.";

cout<<endl<<endl;

break;

case 51: //If choice is 3

//Update data on floor map

updtFloor(roomTp,visitd,inside,floor);

//Display floor map

dispMap(roomTp,floor);

cout<<endl;

break;

case 52: //If choice is 4

return 0; //End Program

break;

default: //If choice is invalid

cout<<"INVALID INPUT"; //Output invalid statment

cout<<endl<<endl;

}

}while(choice!=51); //Reiterate until player leaves

//Move Character

do{

//Display controls

cout<<"CONTROLS"<<endl;

cout<<"'w' = Move forward"<<endl;

cout<<"'s' = Move backward"<<endl;

cout<<"'a' = Move left"<<endl;

cout<<"'d' = Move right"<<endl;

cout<<"Press ENTER after you input your movement.";

cout<<endl<<endl;

cin>>movDir; //Player input

switch(movDir){

case 'w': //If player moves forward

cout<<endl;

//Search for player's current location

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

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

if(inside[i][j]==0){

inside[i][j]=1; //Set current location to empty

inside[i-1][j]=0;//Set location +1 y to filled

visitd[i-1][j]=0;//Set location +1 y to visited

curPosX=j; //Set x coordinate of player

curPosY=i-1; //Set y coordinate of player

//Check for walls or edges

if(i==0||roomTp[floor][i-1][j]=='X'){

//Output failure message

cout<<"There seems to be a wall in the way...";

cout<<endl;

//Set all changed variables back to previous

inside[i][j]=1;

inside[i-1][j]=1;

inside[i+1][j]=0;

visitd[i][j]=1;

}

}

}

}

//Return Original Symbols

fillFloor(roomTp,room,maps);

//Update Map after move

updtFloor(roomTp,visitd,inside,floor);

//Display Map after move

dispMap(roomTp,floor);

break;

case 's': //If player moves backward

cout<<endl;

//Search for player's current location

for(int i=LENGTH-1;i>=0;i--){

for(int j=WIDTH-1;j>=0;j--){

if(inside[i][j]==0){

inside[i][j]=1; //Set current location to empty

inside[i+1][j]=0;//Set location -1 y to filled

visitd[i+1][j]=0;//Set location -1 y to visited

curPosX=j; //Set x coordinate of player

curPosY=i+1; //Set y coordinate of player

//Check for walls or edges

if(i==4||roomTp[floor][i+1][j]=='X'){

//Output failure message

cout<<"There seems to be a wall in the way...";

cout<<endl;

//Set all changed variables back to previous

inside[i][j]=1;

inside[i+1][j]=1;

inside[i-1][j]=0;

visitd[i][j]=1;

}

}

}

}

//Return Original Symbols

fillFloor(roomTp,room,maps);

//Update Map after move

updtFloor(roomTp,visitd,inside,floor);

//Display Map after move

dispMap(roomTp,floor);

break;

case 'a': //If player moves left

cout<<endl;

//Search for player's current location

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

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

if(inside[i][j]==0){

inside[i][j]=1; //Set current location to empty

inside[i][j-1]=0;//Set location -1 x to filled

visitd[i][j-1]=0;//Set location -1 x to visited

curPosX=j-1; //Set x coordinate of player

curPosY=i; //Set y coordinate of player

//Check for walls or edges

if(j==0||roomTp[floor][i][j-1]=='X'){

//Output failure message

cout<<"There seems to be a wall in the way...";

cout<<endl;

//Set all changed variables back to previous

inside[i][j]=1;

inside[i][j-1]=1;

inside[i][j+1]=0;

visitd[i][j]=1;

}

}

}

}

//Return Original Symbols

fillFloor(roomTp,room,maps);

//Update Map after move

updtFloor(roomTp,visitd,inside,floor);

//Display Map after move

dispMap(roomTp,floor);

break;

case 'd': //If player moves right

cout<<endl;

//Search for player's current location

for(int i=LENGTH-1;i>=0;i--){

for(int j=WIDTH-1;j>=0;j--){

if(inside[i][j]==0){

inside[i][j]=1; //Set current location to empty

inside[i][j+1]=0;//Set location -1 x to filled

visitd[i][j+1]=0;//Set location -1 x to visited

curPosX=j+1; //Set x coordinate of player

curPosY=i; //Set y coordinate of player

//Check for walls or edges

if(roomTp[floor][i][j+1]=='X'){

//Output failure message

cout<<"There seems to be a wall in the way...";

cout<<endl;

//Set all changed variables back to previous

inside[i][j]=1;

inside[i][j+1]=1;

inside[i][j-1]=0;

visitd[i][j]=1;

}

}

}

}

//Return Original Symbols

fillFloor(roomTp,room,maps);

//Update Map after move

updtFloor(roomTp,visitd,inside,floor);

//Display Map after move

dispMap(roomTp,floor);

break;

}

cout<<endl;

//Apply effect of tile player is on

if(inside[curPosY][curPosX]==0&&used[curPosY][curPosX]==1){

//Fill floor with original data

fillFloor(roomTp,room,maps);

//Declare XP gain variable

int xpGainz=(rand()%15+1\*(floor+1)); //Gives 1-15 xp\*floor factor

switch(roomTp[floor][curPosY][curPosX]){

case 'E': //If player is on enemy tile

//Start battle

enBattl(gf1,en,enNames,floor,choice,outcome);

if(outcome==1){ //If player loses battle and wants to load

//Load previous state

load(visitd,intro,gf1,floor,data);

//Fill floor with original map data

fillFloor(roomTp,room,maps);

//Set all previously visited rooms to used

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

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

used[i][j]=visitd[i][j];

}

}

//Update ifInsid

ifInsid(roomTp,inside,floor);

//Find boss and assign coordinates

findB(roomTp,floor,bosPosX,bosPosY);

//Spawn player at Hub

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

for(int k=0;k<WIDTH;k++){

if(roomTp[floor][j][k]=='H'){

inside[j][k]=0;

visitd[j][k]=0;

used[j][k]=1;

}

}

}

//Update Map

updtFloor(roomTp,visitd,inside,floor);

//Display Map

dispMap(roomTp,floor);

cout<<endl;

isLoad=1;

}

if(outcome==2){ //If player loses and wants to quit game

return 0;

}

if(outcome==3){ //If player wins battle

//Output amount of XP gained

cout<<gf1.name<<" gained "<<xpGainz<<" EXP";

cout<<endl<<endl;

//Add XP to total

gf1.cxp+=xpGainz;

//Call level up function

lvlUp(gf1);

cout<<endl;

//Output XP status

cout<<"Max XP : "<<gf1.mxp<<endl;

cout<<"Current amount of XP: "<<gf1.cxp<<endl;

cout<<"XP Needed to lvl up : "<<gf1.mxp-gf1.cxp;

cout<<endl<<endl;

//Update Map

updtFloor(roomTp,visitd,inside,floor);

//Display Map

dispMap(roomTp,floor);

cout<<endl;

used[curPosY][curPosX]=0; //Set tile to used

}

break;

case 'T': //If player is on treasure tile

switch(rand()%3){ //Randomly select prize for player

case 0: //Prize 1: Some peace and quiet

cout<<"You scour the room and find nothing of "

<<"value. However, not having to"

<<endl<<"fight one of those abominations is "

<<"a treasure in itself.";

cout<<endl;

used[curPosY][curPosX]=0; //Set tile to used

break;

case 1: //Prize 2: Restore health

cout<<"You find some snack mix in a "

<<"scientist's office. You're not very "

<<endl<<"hungry but "<<gf1.name<<" gladly "

<<"eats it.";

cout<<endl<<endl;

//Set current HP to maximum HP

gf1.chp=gf1.hpt;

//Affirm that health has been restored

cout<<gf1.name<<" has been fully healed!";

cout<<endl;

used[curPosY][curPosX]=0; //Set tile to used

break;

case 2: //Prize 3: Level Up

cout<<"You find some strange pills in a lockbox. "

<<gf1.name<<" consumes them"

<<endl<<"and starts to glow with newfound "

<<"energy.";

cout<<endl<<endl;

//Fill XP meter

gf1.cxp=gf1.mxp;

//Call level up function

lvlUp(gf1);

used[curPosY][curPosX]=0; //Set tile to used

cout<<endl;

//Update Map

updtFloor(roomTp,visitd,inside,floor);

//Display Map

dispMap(roomTp,floor);

cout<<endl;

break;

}

break;

case 'H': //If player is on Hub tile

do{

cout<<"This room appears to still have working "

<<"equipment inside. What would you like to do?";

cout<<endl;

//Prompt user for input

cout<<"1. Save Game"<<endl;

cout<<"2. Rest"<<endl;

cout<<"3. Leave Hub"<<endl;

cout<<"4. Quit Game"<<endl;

cin>>choice; //Input choice

switch(choice){

case 49: //If player chooses to save

//Save Game

save(visitd,intro,gf1,floor,data);

//Affirm game has been saved

cout<<"GAME SAVED"<<endl;

break;

case 50: //If player chooses to rest

//Set current HP to maximum HP

gf1.chp=gf1.hpt;

//Affirm health has been restored

cout<<gf1.name<<" has been fully healed.";

cout<<endl;

break;

case 51: //If player chooses to leave Hub

//Update floor map

updtFloor(roomTp,visitd,inside,floor);

//Display floor map

dispMap(roomTp,floor);

break;

case 52: //If player chooses to quit game

//Exit Program

return 0;

break;

default: //If player input is invalid

//Output failure statement

cout<<"INVALID INPUT";

cout<<endl<<endl;

break;

}

}while(choice!=51); //Reiterate until player leaves Hub

break;

case 'B': //If player lands on boss tile

//Initiate boss battle

bsBattl(gf1,boss,floor,choice,outcome);

if(outcome==1){ //If user loses battle and wants to load

//Load Game

load(visitd,intro,gf1,floor,data);

//Fill floor map with file data

fillFloor(roomTp,room,maps);

//Set all previously visited tiles to used

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

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

used[i][j]=visitd[i][j];

}

}

//Update ifInsid

ifInsid(roomTp,inside,floor);

//Find boss and assign coordinates

findB(roomTp,floor,bosPosX,bosPosY);

//Spawn player at Hub

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

for(int k=0;k<WIDTH;k++){

if(roomTp[floor][j][k]=='H'){

inside[j][k]=0;

visitd[j][k]=0;

used[j][k]=1;

}

}

}

//Update Map

updtFloor(roomTp,visitd,inside,floor);

//Display Map

dispMap(roomTp,floor);

cout<<endl;

isLoad=1; //Set load flag so game can be loaded again

}

if(outcome==2){ //If player loses battle and wants to quit

//End Program

return 0;

break;

}

if(outcome==3){ //If player wins battle

//Output amount of XP gained

cout<<gf1.name<<" gained "<<xpGainz<<" EXP";

cout<<endl<<endl;

//Output amount of additional XP gained

cout<<gf1.name<<" gained an additional "<<xpGainz<<" EXP";

cout<<endl<<endl;

//Add XP

gf1.cxp+=xpGainz;

gf1.cxp+=xpGainz\*3;

//Level Up function

lvlUp(gf1);

cout<<endl;

//Ouput XP status

cout<<"Max XP : "<<gf1.mxp<<endl;

cout<<"Current amount of XP: "<<gf1.cxp<<endl;

cout<<"XP Needed to lvl up : "<<gf1.mxp-gf1.cxp;

cout<<endl<<endl;

//Output floor victory message

cout<<"You walk past "<<boss[floor].name<<"'s corpse "

<<"and proceed onward to the next floor.";

cout<<endl<<endl;

used[curPosY][curPosX]=0; //Set tile to used

}

break;

}

}

//Reiterate until player defeats boss

}while(inside[bosPosX][bosPosY]==1&&visitd[bosPosX][bosPosY]==1);

}

//Exit program

return 0;

Acknowledgements

* Dr. Mark E. Lehr
* Jesus Christ
* Death Wish Coffee Co.