//This is a small game that I have coded in C++ programming language. I invite you to compile it and give it a play. Let me know what you think

#include <iostream> //allows for standard input and output

#include <stdlib.h> //opens the standard library, which includes function rand()

using namespace std; //you wont have to use std::

struct AI { //makes a structure to store enemy information

int x;

int y;

}enemy[99]; //this declares a maximum of 100 enemies, from 0 to 99

struct customboard { //this is a structure to store information about the custom board

int height;

int width;

int enemies;

int lives;

bool winloss;

}custom;

void Print\_Instructions() {

cout<<"\n\nC++ does not offer a graphics library"<<endl;

cout<<"So this game does not use graphics, sorry\n"<<endl;

cout<<"Your avatar is indicated by the number 1"<<endl;

cout<<"An empty space is indicated by the number 0"<<endl;

cout<<"To move your avatar, use keys 2, 4, 6, and 8"<<endl;

cout<<"8 will move you up, 2 will move you down,"<<endl;

cout<<"4 will move you left and 6 will move you right\n"<<endl;

cout<<"Your goal is to get the treasure, indicated by the number 8"<<endl;

cout<<"If you succeed in getting the treasure, you win\n"<<endl;

cout<<"BUT BEWARE! Enemies, indicated by the number 2 will try to stop you"<<endl;

cout<<"Enemies will move one space each turn, chosen at random"<<endl;

cout<<"If an enemy lands on the same square as you, you lose\n"<<endl;

cout<<"A tip to consider: Enemies may be hiding behind the treasure!"<<endl;

}

void InitializeBoardStandard(int board [100][100], customboard &custom, AI enemy[99])

{

int boardx, boardy; //boardx and boardy are the x and y coordinates of the board

for (boardy=0; boardy<10; boardy++) {

for (boardx=0; boardx<10; boardx++) {

board[boardx][boardy]=0; //fills a 10 by 10 board with zeros

}

}

board[0][0]=1; //initial position of the avatar (1)

board[5][5]=2; //initial positions of the enemies (2)

board[2][4]=2;

board[4][2]=2;

board[6][8]=2;

board[8][6]=2;

board[2][2]=2;

board[2][8]=2;

board[8][2]=2;

board[9][9]=8; //position of the treasure (8)

enemy[0].x=5;

enemy[0].y=5;

enemy[1].x=2;

enemy[1].y=4;

enemy[2].x=4;

enemy[2].y=2;

enemy[3].x=6;

enemy[3].y=8;

enemy[4].x=8;

enemy[4].y=6;

enemy[5].x=2;

enemy[5].y=2;

enemy[6].x=2;

enemy[6].y=8;

enemy[7].x=8;

enemy[7].y=2; //these are the x and y coordinates of the first 8 enemies

custom.height=10;

custom.width=10;

custom.enemies=8;

}

void PrintBoardCustom (int board[100][100], AI enemy[99], customboard custom) {

int boardx, boardy;

for (boardy=0; boardy<custom.height; boardy++) {

cout<<"\n"; //goes to the next line after printing 10 values

for (boardx=0; boardx<custom.width; boardx++) {

cout<<board[boardx][boardy]<<" "; //prints the value of the board at that space

}

}

cout<<"\n\n\n\n\n";

}

void ClearBoardCustom(int board[100][100], customboard &custom) {

int boardx, boardy;

for (boardy=0; boardy<custom.height; boardy++) {

for (boardx=0; boardx<custom.width; boardx++) {

board[boardx][boardy]=0; //This function resets the board to a plain 10x10 array of 0s

}

}

}

void RandomPlaceEnemies(int board[100][100], customboard &custom, AI enemy[99]) {

int enemyID;

for (enemyID=0; enemyID<custom.enemies; enemyID++) {

enemy[enemyID].x= rand() % custom.width;

enemy[enemyID].y= rand() % custom.height; //generates a random number for the enemy x and y position

int xplusy;

xplusy=enemy[enemyID].x+enemy[enemyID].y;

while (xplusy % 2!=0 || (enemy[enemyID].x<3 && enemy[enemyID].y<3)) {

//Makes sure the enemies follow two parameters: first that the sum of their x and y coordinates are not odd, as these enemies would be unable to attack the avatar, and secondly that the enemies are not too close to the avatar to start

enemy[enemyID].x= rand() % custom.width;

enemy[enemyID].y= rand() % custom.height;

xplusy=enemy[enemyID].x+enemy[enemyID].y;

}

board[enemy[enemyID].x][enemy[enemyID].y]=2; //place enemies randomly through the board

}

}

void CustomVariables(customboard &custom) {

cout<<"\n\nEnter your board height"<<endl;

cin>>custom.height;

cout<<"Enter your board width"<<endl;

cin>>custom.width;

cout<<"Enter a number of enemies"<<endl;

cin>>custom.enemies; //inputs to build the custom board from

}

void RunGame(int board[100][100], AI enemy[99], customboard &custom) {

int avatarx, avatary; //initializes the position of the avatar

avatarx=0;

avatary=0; //assigns initial position of the avatar

int enemyID;

board[0][0]=1;

board[custom.width-1][custom.height-1]=8;

PrintBoardCustom(board, enemy, custom); //prints the board

while (avatarx!=custom.width - 1 || avatary!=custom.height - 1) {//to make sure you haven't reached the treasure yet!

ClearBoardCustom(board, custom); //clears the board to prevent shadows

cout<<"Move: ";

int avatarMove;

cin>>avatarMove; //to select you move

if (avatary==0 && avatarMove==8) {

avatarMove=2; //keeps the avatar on the board if he tries to go too high

}

if (avatary==(custom.height-1) && avatarMove==2) {

avatarMove=8; //keeps the avatar on the board if he tries to go too low

}

if (avatarx==0 && avatarMove==4) {

avatarMove=6; //keeps the avatar on the board if he tries to go too far right

}

if (avatarx==(custom.width-1) && avatarMove==6) {

avatarMove=4; //keeps the avatar on the board if he tries to go too far left

}

switch(avatarMove) {

case 2:

avatary=avatary+1; //add 1 to your avatar's y coordinate

board[avatarx][avatary]=1; //mark new position of your avatar

break;

case 4:

avatarx=avatarx-1;

board[avatarx][avatary]=1;

break;

case 6:

avatarx=avatarx+1;

board[avatarx][avatary]=1;

break;

case 8:

avatary=avatary-1;

board[avatarx][avatary]=1;

break;

default:

cout<<"Incorrect Input"<<endl;

cout<<"Enter one of the following values:"<<endl;

cout<<"2 to move up"<<endl;

cout<<"4 to move left"<<endl;

cout<<"6 to move right"<<endl;

cout<<"8 to move up"<<endl;

cout<<"Thanks for your consideration"<<endl;

}

if (avatarMove==2 || avatarMove==4 || avatarMove==6 || avatarMove==8) {

//This if statement makes sure the user put a useable input before continueing

int Enemy\_Move;

for (enemyID=0; enemyID<custom.enemies; enemyID++) { //does one enemy move at a time, until it reaches the eighth enemy

Enemy\_Move= rand() % 4 + 1; //chooses a random move for the enemy

int switchenemymove;

switchenemymove=Enemy\_Move+Enemy\_Move; //this line creates symmetry between avatar movement and enemy movement

if (enemy[enemyID].y==0 && switchenemymove==8) {

switchenemymove=2; //keeps the enemy on the board if he tries to go too high

}

if (enemy[enemyID].y==custom.height-1 && switchenemymove==2) {

switchenemymove=8; //keeps the enemy on the board if he tries to go too low

}

if (enemy[enemyID].x==0 && switchenemymove==4) {

switchenemymove=6; //keeps the enemy on the board if he tries to go too far right

}

if (enemy[enemyID].x==custom.width-1 && switchenemymove==6) {

switchenemymove=4; //keeps the enemy on the board if he tries to go too far left

}

if (enemy[enemyID].x==avatarx+1 && enemy[enemyID].y==avatary) {

switchenemymove=4;

}

if (enemy[enemyID].x==avatarx-1 && enemy[enemyID].y==avatary) {

switchenemymove=6;

}

if (enemy[enemyID].y==avatary+1 && enemy[enemyID].x==avatarx) {

switchenemymove=8;

}

if (enemy[enemyID].y==avatary-1 && enemy[enemyID].x==avatarx) {

switchenemymove=2;

}

//The previous four if statements force the enemy to attack the avatar if possible

switch(switchenemymove) {

case 2:

enemy[enemyID].y=enemy[enemyID].y+1;

board[enemy[enemyID].x][enemy[enemyID].y]=2;

break;

case 4:

enemy[enemyID].x=enemy[enemyID].x-1;

board[enemy[enemyID].x][enemy[enemyID].y]=2;

break;

case 6:

enemy[enemyID].x=enemy[enemyID].x+1;

board[enemy[enemyID].x][enemy[enemyID].y]=2;

break;

case 8:

enemy[enemyID].y=enemy[enemyID].y-1;

board[enemy[enemyID].x][enemy[enemyID].y]=2;

break;

} //the switch case is what actually moves the enemy

if ( avatarx==enemy[enemyID].x && avatary==enemy[enemyID].y ) {

cout<<"You Lose!\n\n"<<endl;

custom.winloss=0; //this variable is used in campaign

return; //leave the game if you lost

}

board[custom.width-1][custom.height-1]=8; //make sure an enemy doesn't accidentally run over our treasure indicator

}

PrintBoardCustom(board, enemy, custom); //once all the enemies have moved, print the board

if (avatarx==custom.width-1 && avatary==custom.height-1) { //check if you won

cout<<"Congratulations, You Have Won!\n\n\n"<<endl;

custom.winloss=1;

return;

}

}

}

}

struct constants { //This is to store campaign level information

int height;

int width;

int enemies;

}level[9]; //This initializes 10 levels, from 0 to 9

void RunCampaign(int board[100][100], AI enemy[99], customboard &custom, constants level[9])

{

level[0].height=4;

level[0].width=6;

level[0].enemies=2;

level[1].height=20;

level[1].width=7;

level[1].enemies=10;

level[2].height=16;

level[2].width=21;

level[2].enemies=18;

level[3].height=19;

level[3].width=33;

level[3].enemies=22;

level[4].height=4;

level[4].width=8;

level[4].enemies=3;

level[5].height=7;

level[5].width=11;

level[5].enemies=7;

level[6].height=10;

level[6].width=10;

level[6].enemies=8;

level[7].height=19;

level[7].width=15;

level[7].enemies=17;

level[8].height=5;

level[8].width=15;

level[8].enemies=7;

level[9].height=20;

level[9].width=39;

level[9].enemies=42; //initialize level constants

int difficulty;

cout<<"\n\nEnter 1 for easy difficulty"<<endl;

cout<<"Enter 2 for medium difficulty"<<endl;

cout<<"Enter 3 for hard difficulty"<<endl;

cout<<"Selection: ";

cin>>difficulty;

switch(difficulty) {

case 1:

custom.lives=7;

break;

case 2:

custom.lives=5;

break;

case 3:

custom.lives=3;

break;

default:

custom.lives=3;

break;

}

int levelnumber; //something to mark what level you are on

for (levelnumber=0; levelnumber<10; levelnumber++) { //cycle through levels

custom.height=level[levelnumber].height;

custom.width=level[levelnumber].width;

custom.enemies=level[levelnumber].enemies; //apply level constants

cout<<"\n\nEntering Level "<<(levelnumber + 1)<<endl;

cout<<"Lives: "<<custom.lives<<endl;

ClearBoardCustom(board, custom); //initializes and clears board to begin the game

RandomPlaceEnemies(board, custom, enemy);

RunGame(board, enemy, custom); //where the game is actually run

if (custom.winloss==0) { //opens if you lose the level

custom.lives=custom.lives-1; //lose a life

levelnumber=levelnumber-1; //send you back to the same level

}

if (custom.lives==0) { //if you run out of lives

cout<<"You are out of lives!"<<endl;

cout<<"You reached level "<<(levelnumber+2)<<endl; //print how far you reached

return;

}

if (custom.winloss==1 && levelnumber==9 && difficulty!=3) { //if you beat all 10 levels

cout<<"You have beaten the game\n\n"<<endl;

cout<<"Credits: "<<endl;

cout<<"Drew Loughran -- Level Design Assistant"<<endl;

cout<<"Kevin Loughran -- Debugging Department Head"<<endl;

cout<<"\n????? -- Head Designer"<<endl;

return;

}

if (custom.winloss==1 && levelnumber==9 && difficulty==3) {

cout<<"You have beaten the game\n\n"<<endl;

cout<<"Credits: "<<endl;

cout<<"Drew Loughran -- Level Design Assistant"<<endl;

cout<<"Kevin Loughran -- Debugging Department Head"<<endl;

cout<<"\nBrian Loughran -- Head Designer"<<endl;

return;

}

}

}

int main() {

int board[100][100]; //initializes a board of maximum size 101x101

int selection;

while (selection!=5) { //keeps the menu open if you dont exit

cout<<"1. Start New Game (Standard)"<<endl;

cout<<"2. Start New Game (Custom)"<<endl;

cout<<"3. Instructions"<<endl;

cout<<"4. Campaign"<<endl;

cout<<"5. Exit"<<endl;

cout<<"Enter Your Selection: ";

cin>>selection;

switch (selection) {

case 1:

InitializeBoardStandard(board, custom, enemy);

RunGame(board, enemy, custom);

break;

case 2:

CustomVariables(custom); //function to input variables to customize board

ClearBoardCustom(board, custom); //initializes the custom board

RandomPlaceEnemies(board, custom, enemy);

RunGame(board, enemy, custom); //This allows for board customization, if I ever write the code for it

break;

case 3:

Print\_Instructions(); //prints out some instructions

break;

case 4:

RunCampaign(board, enemy, custom, level); //a series of levels to complete with varying difficulty

case 5:

return(0); //exits the menu and the game

default:

cout<<"\nEnter correct choice\n\n";

}

}

}