#include <queue>

#include "graphics.h"

#include <cstdlib>

#include <ctime>

#include <iostream> // Provides cout

#include <windows.h>

#include<conio.h>

#include<cstdio>

#include "dos.h"

using namespace std;

const int W = 12;

const int EMPTY= 0;

const int BODY = 1;

const int RAND\_POINT= 2;

const int Wall = 3;

const int boardXSize=100;

const int boardYSize=50;

int board[boardXSize][boardYSize]={0};

bool gameover = false;

char msg[50];

int score1=5;

int score2=5;

struct Point{

int x;

int y;

Point(int x,int y): x(x), y(y){}

};

queue<Point> snakeBody;

queue<Point> snakeBody2;

void initscore\_1(int x){

settextstyle(3,0,3);

setcolor(BLACK);

sprintf(msg,"score:%d",x);

outtextxy(510,15,msg);

}

void initscore\_2(int y){

settextstyle(3,0,3);

setcolor(BLACK);

sprintf(msg,"score:%d",y);

outtextxy(615,15,msg);

}

void sou(){

Beep(1000,50);

Beep(1000,100);

Beep(1000,150);

Beep(500,100);

Beep(500,150);

Beep(1000,180);

Beep(1000,150);

Beep(500,100);

Beep(500,150);

Beep(1000,180);

}

/\*

void gameover\_screen(){

initwindow(600,600,"Game Over",100,100);

setbkcolor(2);

cleardevice();

readimagefile("pic.jpg",0,0,600,600);

Beep( 1480,100);

//settextstyle(10,0,6);

//outtextxy(15,110,"Game Over");

//sou();

\_getch();

closegraph();

}

\*/

enum directions{UP,DOWN, LEFT, RIGHT};

directions current\_direction = DOWN;

directions current\_direction2 = DOWN ;

void drawSnakeCell(int x,int y){

setfillstyle(SOLID\_FILL, GREEN);

bar(x\*12+1, y\*12+1, x\*12 + W, y\*12+W);

}

void eraseSnakeCell(int x,int y){

setfillstyle(SOLID\_FILL, WHITE);

bar(x\*12+1, y\*12+1, x\*12 + W, y\*12+W);

}

void drawRandPoint(int x,int y){

setfillstyle(SOLID\_FILL, BLUE);

bar(x\*12+1, y\*12+1, x\*12 + W, y\*12+W);

}

/\*

void START\_screen(){

initwindow(600,600,"Welcome to Snake Game :)",100,100);

setbkcolor(2);

cleardevice();

readimagefile("600x600bb.jpg",0,0,600,600);

//settextstyle(10,0,6);

//outtextxy(15,110,"Game Over");

sou();

delay(500);

closegraph();

}

\*/

void drawWall(int x,int y,colors color,bool full){

setfillstyle(SOLID\_FILL, color);

if(!full){

bar(x\*12+1, y\*12+1, x\*12 + W, y\*12+W);

}else{

bar(x\*12, y\*12, x\*12+W, y\*12+W);

}

}

void draw\_Start\_Screen(){

initwindow(1200, 600,"Snack Game",150,100);

readimagefile("NamesCover.jpg",0,0,1200,600);

delay(1500);

srand (time(NULL));

//setbkcolor(WHITE);

//cleardevice();

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

for(int y=0;y<boardYSize/2;y++){

drawWall(x,y,BLACK,true);

}

delay(1);

}

//Beep(1500,500);

///////////////

for(int x=boardXSize;x>=0;x--){

for(int y=boardYSize;y>=boardYSize/2;y--){

drawWall(x,y,BLACK,true);

}

delay(1);

}

//Beep(1000,1000);

/////////

setbkcolor(BLACK);

cleardevice();

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

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

drawWall(x,y,WHITE,true);

}

delay(1);

}

//Beep(1000,1000);

closegraph();

}

////////////////////////////////////////

void draw\_end\_board(int xStart,int xSize,colors wall\_color){

for(int x=xStart+1;x<xSize-1;x++){

for(int y=1;y<boardYSize-1;y++){

drawWall(x,y,wall\_color,false);

}

delay(1);

}

}

/////////////////////////////

void drawBoard(){

for(int r=0;r<100;r++){

for(int c=0;c<50;c++){

switch(board[r][c]){

case EMPTY:

break;

case BODY:

drawSnakeCell(r,c);

break;

case RAND\_POINT:

drawRandPoint(r,c);

break;

case Wall:

drawWall(r,c,BLACK,false);

break;

}

}

}

}

////////////////

void initWall(){

srand (time(NULL));

for(int x=0;x<100;x++){

for(int y=0;y<50;y++){

if(x==0 || x==49||x==50){

board[x][y] = Wall;

}

else if(y==0 || y==49){

board[x][y] = Wall;

}

}

}

for(int y = 0 ; y <50;y++){

board[99][y]=Wall;

}

}

void initscore\_wall(){

for(int x=40;x<=58;x++){

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

if(y==4){

board[x][y] = Wall;

}

else if (x==40 || x==58){

board[x][y] = Wall;

}

}

}

}

void game\_End\_Screen(int winner){

settextstyle(0, HORIZ\_DIR, 7);

setcolor(WHITE);

if(winner==1){

draw\_end\_board(0,boardXSize/2,GREEN);

setbkcolor(GREEN);

outtextxy(150, 450, "Winner");

outtextxy(100, 100, "PLAYER 1");

draw\_end\_board(boardXSize/2,boardXSize,RED);

setbkcolor(RED);

outtextxy(780, 450, "Loser" );

outtextxy(680,100, "PLAYER 2");

}else{

draw\_end\_board(boardXSize/2,boardXSize,GREEN);

setbkcolor(GREEN);

outtextxy(750, 450, "Winner" );

outtextxy(680,100, "PLAYER 2");

draw\_end\_board(0,boardXSize/2,RED);

setbkcolor(RED);

outtextxy(150, 450, "Loser");

outtextxy(100, 100, "PLAYER 1");

sou();

}

}

void generateRandomPoint(){

int randX, randY;

do{

randX = rand() % 47;

randY = rand() % 49+5;

if (randY>=48 || randY<=6){

randY=25;}

Beep(500,100);

}while(board[randX][randY] != EMPTY);

board[randX][randY] = RAND\_POINT;

drawRandPoint(randX,randY);

}void generateRandomPoint2(){

int randX, randY;

do{

randX = (rand() % 50)+50;

if(randX<=50 || randX>=98){

randX = 70;

}

randY = rand() % 49+5;

if (randY>=48 || randY<=6){

randY=25;}

Beep(250,100);

}while(board[randX][randY] != EMPTY);

board[randX][randY] = RAND\_POINT;

drawRandPoint(randX,randY);

}

void initSnake()

{

srand (time(NULL));

int x = 20;

int y = 5;

for(int i=0;i<6;i++){

board[x][y] = BODY;

Point p(x,y);

snakeBody.push(p);

y += 1;

}

generateRandomPoint();

}

void initSnake2()

{

srand (time(NULL));

int x = 70;

int y = 5;

for(int i=0;i<6;i++){

board[x][y] = BODY;

Point p(x,y);

snakeBody2.push(p);

y += 1;

}

generateRandomPoint2();

}

void moveSnake(directions direction){

current\_direction = direction;

Point p = snakeBody.back();

switch(direction){

case UP:

p.y = p.y - 1;

break;

case DOWN:

p.y = p.y + 1;

break;

case LEFT:

p.x = p.x - 1;

break;

case RIGHT:

p.x = p.x + 1;

break;

}

//خبطه التعبان //////

switch(board[p.x][p.y]){

case EMPTY:

snakeBody.push(p);

drawSnakeCell(p.x,p.y);

board[p.x][p.y] = BODY;

p = snakeBody.front();

snakeBody.pop();

eraseSnakeCell(p.x,p.y);

board[p.x][p.y] = EMPTY;

break;

case RAND\_POINT:

snakeBody.push(p);

drawSnakeCell(p.x,p.y);

board[p.x][p.y] = BODY;

score1=score1-1;

initscore\_1(score1);

generateRandomPoint();

if (score1 ==0){

game\_End\_Screen(1);

gameover=true;

}

break;

case Wall:

//setbkcolor(BLACK);

//cleardevice();

game\_End\_Screen(2);

gameover=true;

break;

case BODY:

//setbkcolor(BLACK);

//cleardevice();

game\_End\_Screen(2);

gameover=true;

break;

}

}

void moveSnake2(directions direction){

current\_direction2 = direction;

Point p = snakeBody2.back();

switch(direction){

case UP:

p.y = p.y - 1;

break;

case DOWN:

p.y = p.y + 1;

break;

case LEFT:

p.x = p.x - 1;

break;

case RIGHT:

p.x = p.x + 1;

break;

}

//خبطه التعبان //////

switch(board[p.x][p.y]){

case EMPTY:

snakeBody2.push(p);

drawSnakeCell(p.x,p.y);

board[p.x][p.y] = BODY;

p = snakeBody2.front();

snakeBody2.pop();

eraseSnakeCell(p.x,p.y);

board[p.x][p.y] = EMPTY;

break;

case RAND\_POINT:

snakeBody2.push(p);

drawSnakeCell(p.x,p.y);

board[p.x][p.y] = BODY;

score2=score2-1;

initscore\_2(score2);

generateRandomPoint2();

if (score2 ==0){

game\_End\_Screen(2);

gameover=true;

}

break;

case Wall:

game\_End\_Screen(1);

gameover=true;

break;

case BODY:

game\_End\_Screen(1);

gameover=true;

break;

}

}

void game(){

draw\_Start\_Screen();

int i;

char c;

initwindow(1200, 600,"Snack Game",100,100);

setbkcolor(WHITE);

cleardevice();

initSnake();

initSnake2();

initWall();

initscore\_wall();

initscore\_1(score1);

initscore\_2(score2);

drawBoard();

clock\_t start = clock();

clock\_t end;

do

{

//delay(100);

end = clock();

int diff = end - start;

if(diff >= 200){

if(gameover!=true){

moveSnake(current\_direction);

moveSnake2(current\_direction2);

}

start = clock();

}

if(kbhit()){

c = (char) getch( );

if (c != 0)

switch (c)

{

case 119://w

if(current\_direction!=DOWN && gameover!= true&&current\_direction!=UP){

moveSnake(UP);

}

break;

case 97://a

if(current\_direction!=RIGHT&& gameover!= true&&current\_direction!=LEFT){

moveSnake(LEFT);

}

break;

case 100://d

if(current\_direction!=LEFT&& gameover!= true&&current\_direction!=RIGHT){

moveSnake(RIGHT);

}

break;

case 115://s

if(current\_direction!=UP&& gameover!= true&&current\_direction!=DOWN){

moveSnake(DOWN);

}

break;

default: cout << "Unknown extended key." << endl;

}

else

{ // Process one of the special keys:

c = (char)getch();

switch (c)

{

case KEY\_UP:

if(current\_direction2!=DOWN && gameover!= true&&current\_direction2!=UP){

moveSnake2(UP);

}

break;

case KEY\_LEFT:

if(current\_direction2!=RIGHT&& gameover!= true&&current\_direction2!=LEFT){

moveSnake2(LEFT);

}

break;

case KEY\_RIGHT:

if(current\_direction2!=LEFT&& gameover!= true&&current\_direction2!=RIGHT){

moveSnake2(RIGHT);

}

break;

case KEY\_DOWN:

if(current\_direction2!=UP&& gameover!= true&&current\_direction2!=DOWN){

moveSnake2(DOWN);

}

break;

default: cout << "Unknown extended key." << endl;

}

}

}

}while (c != 27);

while( ! kbhit());

closegraph( );

}

int main( )

{

game();

}