using System;

using System.Collections.Generic;

using System.Diagnostics;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

namespace ConsoleApp1 {

class Program {

static readonly int gridW = 90;

static readonly int gridH = 25;

static Cell[, ] grid = new Cell[gridH, gridW];

static Cell currentCell;

static Cell food;

static int FoodCount;

static int direction; //0=Up 1=Right 2=Down 3=Left

static readonly int speed = 1;

static bool Populated = false;

static bool Lost = false;

static int snakeLength;

static void Main(string[] args) {

if (!Populated) {

FoodCount = 0;

snakeLength = 5;

populateGrid();

currentCell = grid[(int) Math.Ceiling((double) gridH / 2), (int) Math.Ceiling((double) gridW / 2)];

updatePos();

addFood();

Populated = true;

}

while (!Lost) {

Restart();

}

}

static void Restart() {

Console.SetCursorPosition(0, 0);

printGrid();

Console.WriteLine("Length: {0}", snakeLength);

getInput();

}

static void updateScreen() {

Console.SetCursorPosition(0, 0);

printGrid();

Console.WriteLine("Length: {0}", snakeLength);

}

static void getInput() {

//Console.Write("Where to move? [WASD] ");

ConsoleKeyInfo input;

while (!Console.KeyAvailable) {

Move();

updateScreen();

}

input = Console.ReadKey();

doInput(input.KeyChar);

}

static void checkCell(Cell cell) {

if (cell.val == "%") {

eatFood();

}

if (cell.visited) {

Lose();

}

}

static void Lose() {

Console.WriteLine("\n You lose!");

Thread.Sleep(1000);

Process.Start(System.Reflection.Assembly.GetExecutingAssembly().Location);

Environment.Exit(-1);

}

static void doInput(char inp) {

switch (inp) {

case 'w':

goUp();

break;

case 's':

goDown();

break;

case 'a':

goRight();

break;

case 'd':

goLeft();

break;

}

}

static void addFood() {

Random r = new Random();

Cell cell;

while (true) {

cell = grid[r.Next(grid.GetLength(0)), r.Next(grid.GetLength(1))];

if (cell.val == " ")

cell.val = "%";

break;

}

}

static void eatFood() {

snakeLength += 1;

addFood();

}

static void goUp() {

if (direction == 2)

return;

direction = 0;

}

static void goRight() {

if (direction == 3)

return;

direction = 1;

}

static void goDown() {

if (direction == 0)

return;

direction = 2;

}

static void goLeft() {

if (direction == 1)

return;

direction = 3;

}

static void Move() {

if (direction == 0) {

//up

if (grid[currentCell.y - 1, currentCell.x].val == "\*") {

Lose();

return;

}

visitCell(grid[currentCell.y - 1, currentCell.x]);

} else if (direction == 1) {

//right

if (grid[currentCell.y, currentCell.x - 1].val == "\*") {

Lose();

return;

}

visitCell(grid[currentCell.y, currentCell.x - 1]);

} else if (direction == 2) {

//down

if (grid[currentCell.y + 1, currentCell.x].val == "\*") {

Lose();

return;

}

visitCell(grid[currentCell.y + 1, currentCell.x]);

} else if (direction == 3) {

//left

if (grid[currentCell.y, currentCell.x + 1].val == "\*") {

Lose();

return;

}

visitCell(grid[currentCell.y, currentCell.x + 1]);

}

Thread.Sleep(speed \* 100);

}

static void visitCell(Cell cell) {

currentCell.val = "#";

currentCell.visited = true;

currentCell.decay = snakeLength;

checkCell(cell);

currentCell = cell;

updatePos();

//checkCell(currentCell);

}

static void updatePos() {

currentCell.Set("@");

if (direction == 0) {

currentCell.val = "^";

} else if (direction == 1) {

currentCell.val = "<";

} else if (direction == 2) {

currentCell.val = "v";

} else if (direction == 3) {

currentCell.val = ">";

}

currentCell.visited = false;

return;

}

static void populateGrid() {

Random random = new Random();

for (int col = 0; col < gridH; col++) {

for (int row = 0; row < gridW; row++) {

Cell cell = new Cell();

cell.x = row;

cell.y = col;

cell.visited = false;

if (cell.x == 0 || cell.x > gridW - 2 || cell.y == 0 || cell.y > gridH - 2)

cell.Set("\*");

else

cell.Clear();

grid[col, row] = cell;

}

}

}

static void printGrid() {

string toPrint = "";

for (int col = 0; col < gridH; col++) {

for (int row = 0; row < gridW; row++) {

grid[col, row].decaySnake();

toPrint += grid[col, row].val;

}

toPrint += "\n";

}

Console.WriteLine(toPrint);

}

public class Cell {

public string val {

get;

set;

}

public int x {

get;

set;

}

public int y {

get;

set;

}

public bool visited {

get;

set;

}

public int decay {

get;

set;

}

public void decaySnake() {

decay -= 1;

if (decay == 0) {

visited = false;

val = " ";

}

}

public void Clear() {

val = " ";

}

public void Set(string newVal) {

val = newVal;

}

}

}

}