МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение

высшего образования

**«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ СИСТЕМ И ТЕХНОЛОГИЙ

Кафедра «Информационные системы»

Дисциплина «Основы алгоритмизации и программирования»

Лабораторная работа №8

Визуализация двухмерного массива.

Игра на основе двухмерного массива.

Выполнил:

студент гр. ПИбд-12

Дозорова Алена

Ульяновск, 2021 г

Задание лабораторной работы

1. Создать модуль Game.h / Game.cpp. Перенести в него всю функциональность игры.
2. При нажатии клавиши ‘L’ слева от игрока вставить элемент стены.
3. При нажатии клавиши ‘R’ непосредственно справа от игрока вставить элемент стены.
4. При нажатии клавиши ‘Z’ уничтожить элементы стен начиная справа от игрока и до правой границы игрового поля.
5. При нажатии клавиши ‘A’ уничтожить все элементы стен, которых непосредственно касается игрок – со всех сторон!
6. При нажатии клавиши ‘O’ уничтожить все элементы стен со всех сторон на расстояние 2!
7. При нажатии клавиши ‘D’ в столбце, где больше всего элементов стен, удалить самый верхний элемент стены.
8. Добавить еще 3-5 действий при нажатии на выбранные вами клавиши!

Код программы #pragma once

#define \_CRT\_SECURE\_NO\_WARNINGS

#include "stdio.h"

#define WIN32\_LEAN\_AND\_MEAN

#include <stdlib.h>

#include "framework.h"

// Файлы заголовков среды выполнения C

#include <stdlib.h>

#include <malloc.h>

#include <memory.h>

#include <tchar.h>

#define N 10

#define M 15

int a[N][M] = {

{ 3, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },

{ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },

{ 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 },

{ 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0 },

{ 0, 0, 0, 0, 2, 2, 2, 2, 2, 2, 0, 2, 0, 0, 0 },

{ 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 2, 0, 0, 0 },

{ 0, 0, 1, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0 },

{ 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 2, 0, 0, 0 },

{ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0 },

{ 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0 }

};

int steps = 0;

int gold = 0;

void moveToLeft() {

int i, j;

i = 0;

while (i < N) {

j = 1;

while (j < M) {

if (a[i][j] == 3) {

if (a[i][j - 1] == 0) {

a[i][j - 1] = 3;

a[i][j] = 0;

steps++;

}

else if (a[i][j - 1] == 1) {

a[i][j - 1] = 3;

a[i][j] = 0;

steps++;

gold++;

}

}

j++;

}

i++;

}

}

void moveToRight() {

int i = 0;

while (i < N) {

int j = M - 2;

while (j >= 0) {

if (a[i][j] == 3) {

if (a[i][j + 1] == 0) {

a[i][j + 1] = 3;

a[i][j] = 0;

steps++;

}

else if (a[i][j + 1] == 1) {

a[i][j + 1] = 3;

a[i][j] = 0;

steps++;

gold++;

}

}

j--;

}

i++;

}

}

void moveUp() {

int i = 1;

while (i < N) {

int j = 0;

while (j < M) {

if (a[i][j] == 3) {

if (a[i - 1][j] == 0) {

a[i - 1][j] = 3;

a[i][j] = 0;

steps++;

}

else if (a[i - 1][j] == 1) {

a[i - 1][j] = 3;

a[i][j] = 0;

steps++;

gold++;

}

}

j++;

}

i++;

}

}

void moveDown() {

int i = N-2;

while (i >= 0) {

int j = 0;

while (j < M) {

if (a[i][j] == 3) {

if (a[i + 1][j] == 0) {

a[i + 1][j] = 3;

a[i][j] = 0;

steps++;

}

else if (a[i + 1][j] == 1) {

a[i + 1][j] = 3;

a[i][j] = 0;

steps++;

gold++;

}

}

j++;

}

i--;

}

}

void Create\_wall\_left()

{

int i, j;

i = 0;

while (i < N) {

j = 1;

while (j < M) {

if (a[i][j] == 3) {

if (a[i][j - 1] == 0) {

a[i][j - 1] = 2;

}

}

j++;

}

i++;

}

}

void Create\_wall\_right()

{

int i, j;

i = 0;

while (i < N) {

j = 0;

while (j < M) {

if (a[i][j] == 3) {

if (a[i][j + 1] == 0) {

a[i][j + 1] = 2;

}

}

j++;

}

i++;

}

}

void Destroy\_wall\_right()

{

int i, j, pi, pj;

i = 0;

while (i < N) {

j = 0;

while (j < M) {

if (a[i][j] == 3) {

pi = i;

pj = j;

break;

}

j++;

}

i++;

}

while (pj < M)

{

if (a[pi][pj] == 2)

{

a[pi][pj] = 0;

}

pj+= 1;

}

}

void Destroy\_wall(int ras)

{

int i, j;

i = 1;

while (i < N) {

j = 1;

while (j < M) {

if (a[i][j] == 3) {

if (a[i][j + ras] == 2) {

a[i][j + ras] = 0;

}

if (a[i][j - ras] == 2) {

a[i][j - ras] = 0;

}

if (a[i+ras][j] == 2) {

a[i+ras][j] = 0;

}

if (a[i-ras][j] == 2) {

a[i-ras][j] = 0;

}

}

j++;

}

i++;

}

}

void Destroy\_wall\_D()

{

int i, j, n;

j = 0;

int maxn = 0;

int mi = 0;

while (j < M) {

i = 0;

n = 0;

while (i < N) {

if (a[i][j] == 2) {

n += 1;

}

i++;

}

if (n > maxn)

{

maxn = n;

mi = j;

}

j++;

}

i = 0;

while (i < N) {

if (a[i][mi] == 2) {

a[i][mi] = 0;

break;

}

i++;

}

}

void Double() {

gold \*= 2;

}

void Again()

{

gold = 0;

steps = 0;

int i = 0;

while (i < N) {

int j = 0;

while (j < M) {

if (a[i][j] == 3) {

a[i][j] = 0;

a[0][0] = 3;

}

j++;

}

i++;

}

}

void to\_end()

{

int i = 0;

while (i < N) {

int j = 0;

while (j < M) {

if (a[i][j] == 3) {

a[i][j] = 0;

a[N-1][M-1] = 3;

}

j++;

}

i++;

}

}

int sizeX = 36;

int sizeY = 30;

void DrawField(HDC hdc) {

HBRUSH hBrushEmptyCell; //создаём кисть для пустого поля

hBrushEmptyCell = CreateSolidBrush(RGB(200, 200, 200)); // серый

HBRUSH hBrushGold; //создаём кисть для поля с золотом

hBrushGold = CreateSolidBrush(RGB(255, 255, 0)); // желтый

HBRUSH hBrushWall; //создаём кисть для стены

hBrushWall = CreateSolidBrush(RGB(0, 0, 0)); // черный

HBRUSH hBrushMan; //создаём кисть для игрока

hBrushMan = CreateSolidBrush(RGB(0, 0, 255)); // синий

int i, j;

i = 0;

while (i < N) {

j = 0;

while (j < M) {

RECT rect = { j \* sizeX, i \* sizeY, (j + 1) \* sizeX,(i + 1) \* sizeY };

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

FillRect(hdc, &rect, hBrushEmptyCell);

}

else if (a[i][j] == 1) {

FillRect(hdc, &rect, hBrushGold);

}

else if (a[i][j] == 2) {

FillRect(hdc, &rect, hBrushWall);

}

else if (a[i][j] == 3) {

FillRect(hdc, &rect, hBrushMan);

}

else {

// тут никогда не должны оказаться

}

j++;

}

i++;

}

HFONT hFont;

hFont = CreateFont(20,

0, 0, 0, 0, 0, 0, 0,

DEFAULT\_CHARSET,

0, 0, 0, 0,

L"Courier New"

);

SelectObject(hdc, hFont);

SetTextColor(hdc, RGB(0, 128, 128));

TCHAR string1[] = \_T("сделано ходов:");

TCHAR string2[] = \_T("собрано золота:");

TextOut(hdc, 10, sizeY \* (N + 1), (LPCWSTR)string1, \_tcslen(string1));

TextOut(hdc, 10, sizeY \* (N + 1) + 20, (LPCWSTR)string2, \_tcslen(string2));

char sSteps[5];

TCHAR tsSteps[5];

sprintf(sSteps, "%d", steps);

OemToChar(sSteps, tsSteps);

TextOut(hdc, 220, sizeY \* (N + 1), (LPCWSTR)tsSteps, \_tcslen(tsSteps));

char sGold[5];

TCHAR tsGold[5];

sprintf(sGold, "%d", gold);

OemToChar(sGold, tsGold);

TextOut(hdc, 220, sizeY \* (N + 1) + 20, (LPCWSTR)tsGold, \_tcslen(tsGold));

DeleteObject(hFont);

DeleteObject(hBrushEmptyCell);

DeleteObject(hBrushGold);

DeleteObject(hBrushWall);

DeleteObject(hBrushMan);

}

case WM\_KEYDOWN:

switch (wParam)

{

case VK\_DOWN:

moveDown();

InvalidateRect(hWnd, NULL, TRUE);

break;

case VK\_LEFT:

moveToLeft();

InvalidateRect(hWnd, NULL, TRUE);

break;

case VK\_UP:

moveUp();

InvalidateRect(hWnd, NULL, TRUE);

break;

case VK\_RIGHT:

moveToRight();

InvalidateRect(hWnd, NULL, TRUE);

break;

case 76:

Create\_wall\_left();

InvalidateRect(hWnd, NULL, TRUE);

break;

case 82:

Create\_wall\_right();

InvalidateRect(hWnd, NULL, TRUE);

break;

case 90:

Destroy\_wall\_right();

InvalidateRect(hWnd, NULL, TRUE);

break;

case 65:

Destroy\_wall(1);

InvalidateRect(hWnd, NULL, TRUE);

break;

case 79:

Destroy\_wall(2);

InvalidateRect(hWnd, NULL, TRUE);

break;

case 68:

Destroy\_wall\_D();

InvalidateRect(hWnd, NULL, TRUE);

break;

case 88:

Double();

InvalidateRect(hWnd, NULL, TRUE);

break;

case 48:

Again();

InvalidateRect(hWnd, NULL, TRUE);

break;

case 49:

to\_end();

InvalidateRect(hWnd, NULL, TRUE);

break;

}

Break;