Листинг программы

# Файл h

#pragma once

#include "resource.h"

#define FILE\_PATH \_T("data.dat")

HBITMAP hBitmapNum0=(HBITMAP)LoadImage(NULL, \_T("data//BKG.BMP"), IMAGE\_BITMAP,0,0, LR\_LOADFROMFILE);

HBITMAP hBitmapNum1=(HBITMAP)LoadImage(NULL, \_T("data//BALLS1.BMP"), IMAGE\_BITMAP,0,0, LR\_LOADFROMFILE);

HBITMAP hBitmapNum2=(HBITMAP)LoadImage(NULL, \_T("data//BALLS2.BMP"), IMAGE\_BITMAP,0,0, LR\_LOADFROMFILE);

HBITMAP hBitmapNum3=(HBITMAP)LoadImage(NULL, \_T("data//BALLS3.BMP"), IMAGE\_BITMAP,0,0, LR\_LOADFROMFILE);

HBITMAP hBitmapNum4=(HBITMAP)LoadImage(NULL, \_T("data//BALLS4.BMP"), IMAGE\_BITMAP,0,0, LR\_LOADFROMFILE);

HBITMAP hBitmapNum5=(HBITMAP)LoadImage(NULL, \_T("data//BALLS5.BMP"), IMAGE\_BITMAP,0,0, LR\_LOADFROMFILE);

HBITMAP hBitmapNum6=(HBITMAP)LoadImage(NULL, \_T("data//BALLS6.BMP"), IMAGE\_BITMAP,0,0, LR\_LOADFROMFILE);

HBITMAP hBitmapNum7=(HBITMAP)LoadImage(NULL, \_T("data//BALLS7.BMP"), IMAGE\_BITMAP,0,0, LR\_LOADFROMFILE);

HBITMAP hBitmapNum8=(HBITMAP)LoadImage(NULL, \_T("data//TIME.BMP"), IMAGE\_BITMAP,0,0, LR\_LOADFROMFILE);

HBITMAP hBitmapNum9=(HBITMAP)LoadImage(NULL, \_T("data//NEXT.BMP"), IMAGE\_BITMAP,0,0, LR\_LOADFROMFILE);

HBITMAP hBitmapNum10=(HBITMAP)LoadImage(NULL, \_T("data//DIGITS.BMP"), IMAGE\_BITMAP,0,0, LR\_LOADFROMFILE);

HBITMAP hBitmapNum11=(HBITMAP)LoadImage(NULL, \_T("data//DIGTR.BMP"), IMAGE\_BITMAP,0,0, LR\_LOADFROMFILE);

HBITMAP hBitmapNum12=(HBITMAP)LoadImage(NULL, \_T("data//TIMTR.BMP"), IMAGE\_BITMAP,0,0, LR\_LOADFROMFILE);

HBITMAP hBitmapNum13=(HBITMAP)LoadImage(NULL, \_T("data//NEXTTR.BMP"), IMAGE\_BITMAP,0,0, LR\_LOADFROMFILE);

TCHAR MY\_SOUND0[]=\_T("data//move.wav");

TCHAR MY\_SOUND1[]=\_T("data//destroy.wav");

TCHAR MY\_SOUND2[]=\_T("data//fetus.wav");

TCHAR MY\_SOUND3[]=\_T("data//cantmove.wav");

typedef struct

{

BYTE hour,min,sec;

}TIME;

const int walkability\_fetus=1376256;

const int zero=0x0000ffff;

const int mapWidth = 9, mapHeight = 9;

int onClosedList = 10;

const int notfinished = 0, notStarted = 0;

const int found = 1, nonexistent = 2;

const int walkable = 0;

int walkability [mapWidth][mapHeight];

int openList[mapWidth\*mapHeight+2];

int whichList[mapWidth+1][mapHeight+1];

int openX[mapWidth\*mapHeight+2];

int openY[mapWidth\*mapHeight+2];

int parentX[mapWidth+1][mapHeight+1];

int parentY[mapWidth+1][mapHeight+1];

int Fcost[mapWidth+1][mapHeight+1];

int pathLength;

POINT \*pathBank;

static int path=0;

int xPath;

int yPath;

//-----------------------------------------------------------------------------

// Реализация алгоритма A\*(A-star)

//-----------------------------------------------------------------------------

int FindPath (int startX, int startY, int targetX, int targetY)

{

int onOpenList=0, parentXval=0, parentYval=0,

a=0, b=0, m=0, u=0, v=0, temp=0, numberOfOpenListItems=0,

path = 0, x=0, y=0,

tempx, pathX, pathY, cellPosition,

newOpenListItemID=0;

if (startX == targetX && startY == targetY)

return found;

if (walkability[targetX][targetY] != walkable &&

HIWORD(walkability [targetX][targetY])!=HIWORD(walkability\_fetus))

goto noPath;

for (x = 0; x < mapWidth; x++)

{

for (y = 0; y < mapHeight;y++)

whichList [x][y] = 0;

}

onClosedList = 2;

onOpenList = 1;

pathLength = notStarted;

numberOfOpenListItems = 1;

openList[1] = 1;

openX[1] = startX;

openY[1] = startY;

do

{

if (numberOfOpenListItems != 0)

{

parentXval = openX[openList[1]];

parentYval = openY[openList[1]];

whichList[parentXval][parentYval] = onClosedList;

numberOfOpenListItems--;

openList[1] = openList[numberOfOpenListItems+1];

v = 1;

do

{

u = v;

if (2\*u+1 <= numberOfOpenListItems)

{

if (Fcost[openX[openList[u]]][openY[openList[u]]]>=

Fcost[openX[openList[2\*u]]][openY[openList[2\*u]]])

v = 2\*u;

if (Fcost[openX[openList[v]]][openY[openList[v]]]>=

Fcost[openX[openList[2\*u+1]]][openY[openList[2\*u+1]]])

v = 2\*u+1;

}

else

{

if (2\*u <= numberOfOpenListItems)

{

if (Fcost[openX[openList[u]]][openY[openList[u]]]>=

Fcost[openX[openList[2\*u]]][openY[openList[2\*u]]])

v = 2\*u;

}

}

if (u != v)

{

temp = openList[u];

openList[u] = openList[v];

openList[v] = temp;

}

else

break;

}

while (1);

for (a = parentXval-1; a <= parentXval+1; a++){

for (b = parentYval-1+abs(parentXval-a); b <= parentYval+1-abs(parentXval-a); b++){

if (a != -1 && b != -1 && a != mapWidth && b != mapHeight){

if (whichList[a][b] != onClosedList) {

if (walkability [a][b] == walkable || HIWORD(walkability [a][b])==HIWORD(walkability\_fetus)) {

if (whichList[a][b] != onOpenList)

{

newOpenListItemID++;

m = numberOfOpenListItems+1;

openList[m] = newOpenListItemID;

openX[newOpenListItemID] = a;

openY[newOpenListItemID] = b;

Fcost[a][b] = 10\*(abs(a - targetX) + abs(b - targetY));

parentX[a][b] = parentXval;

parentY[a][b] = parentYval;

while (m != 1)

{

if (Fcost[openX[openList[m]]][openY[openList[m]]] <=

Fcost[openX[openList[m/2]]][openY[openList[m/2]]])

{

temp = openList[m/2];

openList[m/2] = openList[m];

openList[m] = temp;

m = m/2;

}

else

break;

}

numberOfOpenListItems++;

whichList[a][b] = onOpenList;

}

else

{

if (Fcost[parentXval][parentYval] < Fcost[a][b])

{

parentX[a][b] = parentXval;

parentY[a][b] = parentYval;

for (int x = 1; x <= numberOfOpenListItems; x++)

{

if (openX[openList[x]] == a && openY[openList[x]] == b)

{

m = x;

while (m != 1)

{

if (Fcost[openX[openList[m]]][openY[openList[m]]] <

Fcost[openX[openList[m/2]]][openY[openList[m/2]]])

{

temp = openList[m/2];

openList[m/2] = openList[m];

openList[m] = temp;

m = m/2;

}

else

break;

}

break;

}

}

}

}

}

}

}

}

}

}

else

{

path = nonexistent; break;

}

if (whichList[targetX][targetY] == onOpenList)

{

path = found; break;

}

}

while (1);

if (path == found)

{

pathX = targetX; pathY = targetY;

do

{

tempx = parentX[pathX][pathY];

pathY = parentY[pathX][pathY];

pathX = tempx;

pathLength = pathLength + 1;

}

while (pathX != startX || pathY != startY);

pathBank = (POINT\*) realloc (pathBank,

pathLength\*sizeof(POINT));

pathX = targetX ; pathY = targetY;

cellPosition = pathLength;

do

{

cellPosition--;

pathBank [cellPosition].x = pathX;

pathBank [cellPosition].y = pathY;

tempx = parentX[pathX][pathY];

pathY = parentY[pathX][pathY];

pathX = tempx;

}

while (pathX != startX || pathY != startY);

}

return path;

noPath:

xPath = startX;

yPath = startY;

return nonexistent;

}

void EndPathfinder (void)

{

if(pathBank)

{

free (pathBank);

pathBank=NULL;

}

}

POINT \* len(int StartX, int StartY,int &length)

{

int north=1,south=1,west=1,east=1,width=0,height=0;

POINT \*PatchDestroy=NULL;

length=0;

while(StartY-north!=-1 && walkability[StartX][StartY]==walkability[StartX][StartY-north])

north++;

while(StartY+south!=mapHeight && walkability[StartX][StartY]==walkability[StartX][StartY+south])

south++;

while(StartX-west!=-1 && walkability[StartX][StartY]==walkability[StartX-west][StartY])

west++;

while(StartX+east!=mapWidth && walkability[StartX][StartY]==walkability[StartX+east][StartY])

east++;

height=north+south-1;

width=west+east-1;

if(height>=5 && width>=5 && height>=width || height>=5)

{

length=height;

PatchDestroy=(POINT\*)malloc(sizeof(POINT)\*height);

StartY-=north-1;

for (int i=0;i<height;i++)

{

PatchDestroy[i].x=StartX;

PatchDestroy[i].y=StartY+i;

}

}

else

if(width>=5)

{

length=width;

PatchDestroy=(POINT\*)malloc(sizeof(POINT)\*width);

StartX-=west-1;

for (int i=0;i<width;i++)

{

PatchDestroy[i].x=StartX+i;

PatchDestroy[i].y=StartY;

}

}

return PatchDestroy;

}

void dest\_len(POINT \* PatchDestroy)

{

if(PatchDestroy)

{

free(PatchDestroy);

PatchDestroy=NULL;

}

}

# Файл cpp

#include "stdafx.h"

#include "5.0.h"

#include <commctrl.h>

#include <ctime>

#include <mmsystem.h>

#define MAX\_LOADSTRING 100

#define TEMER\_TIME 2

// Global Variables:

HINSTANCE hInst; // current instance

TCHAR szTitle[MAX\_LOADSTRING]; // The title bar text

TCHAR szWindowClass[MAX\_LOADSTRING]; // the main window class name

HIMAGELIST himl,himl2,himl3,himl4;

DWORD T1,T2;

TIME my\_time={0};

int startX, startY;

POINT fetus[3];

BOOL START=FALSE;

int SCORE=0, COUNT\_BUBBLES=0;

// Forward declarations of functions included in this code module:

ATOM MyRegisterClass(HINSTANCE hInstance);

BOOL InitInstance(HINSTANCE, int);

LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);

INT\_PTR CALLBACK About(HWND, UINT, WPARAM, LPARAM);

void Paint(HDC hdc,HDC hdcMem)

{

BITMAP bm;

HDC hDC=CreateCompatibleDC (hdc);

SelectObject (hDC, (HBITMAP)hBitmapNum0);

GetObject (hBitmapNum0, sizeof(bm), &bm);

BitBlt (hdcMem,0,0,bm.bmWidth,bm.bmHeight,hDC,0,0,SRCCOPY);

if(START)

for(int i=0;i<3;i++)

{

ImageList\_Draw(himl3,LOWORD(walkability[fetus[i].x][fetus[i].y])/22

,hdcMem,163+i\*25,5,ILD\_TRANSPARENT);

}

int mul=10000;

for(int i=0;i<5;i++)

{

ImageList\_Draw(himl4,(SCORE / mul) %10,hdcMem,290+i\*20,8,ILD\_TRANSPARENT);

mul/=10;

}

ImageList\_Draw(himl2,my\_time.hour/10,hdcMem,174,34,ILD\_TRANSPARENT);

ImageList\_Draw(himl2,my\_time.hour%10,hdcMem,183,34,ILD\_TRANSPARENT);

ImageList\_Draw(himl2,my\_time.min/10,hdcMem,193,34,ILD\_TRANSPARENT);

ImageList\_Draw(himl2,my\_time.min%10,hdcMem,202,34,ILD\_TRANSPARENT);

ImageList\_Draw(himl2,my\_time.sec/10,hdcMem,212,34,ILD\_TRANSPARENT);

ImageList\_Draw(himl2,my\_time.sec%10,hdcMem,220,34,ILD\_TRANSPARENT);

for (int i=0; i<9; i++)

for (int j=0; j<9; j++)

if (walkability[i][j])

ImageList\_Draw(himl,LOWORD(walkability[i][j])+HIWORD(walkability[i][j])-1,hdcMem,45\*i+5,45\*j+56,ILD\_NORMAL);

DeleteDC(hDC);

}

int APIENTRY \_tWinMain(HINSTANCE hInstance,

HINSTANCE hPrevInstance,

LPTSTR lpCmdLine,

int nCmdShow)

{

UNREFERENCED\_PARAMETER(hPrevInstance);

UNREFERENCED\_PARAMETER(lpCmdLine);

// TODO: Place code here.

MSG msg;

HACCEL hAccelTable;

// Initialize global strings

LoadString(hInstance, IDS\_APP\_TITLE, szTitle, MAX\_LOADSTRING);

LoadString(hInstance, IDC\_MY50, szWindowClass, MAX\_LOADSTRING);

MyRegisterClass(hInstance);

// Perform application initialization:

if (!InitInstance (hInstance, nCmdShow))

{

return FALSE;

}

hAccelTable = LoadAccelerators(hInstance, MAKEINTRESOURCE(IDC\_MY50));

// Main message loop:

while (GetMessage(&msg, NULL, 0, 0))

{

if (!TranslateAccelerator(msg.hwnd, hAccelTable, &msg))

{

TranslateMessage(&msg);

DispatchMessage(&msg);

}

}

return (int) msg.wParam;

}

//

// FUNCTION: MyRegisterClass()

//

// PURPOSE: Registers the window class.

//

// COMMENTS:

//

// This function and its usage are only necessary if you want this code

// to be compatible with Win32 systems prior to the 'RegisterClassEx'

// function that was added to Windows 95. It is important to call this function

// so that the application will get 'well formed' small icons associated

// with it.

//

ATOM MyRegisterClass(HINSTANCE hInstance)

{

WNDCLASSEX wcex;

wcex.cbSize = sizeof(WNDCLASSEX);

wcex.style = CS\_HREDRAW | CS\_VREDRAW;

wcex.lpfnWndProc = WndProc;

wcex.cbClsExtra = 0;

wcex.cbWndExtra = 0;

wcex.hInstance = hInstance;

wcex.hIcon = (HICON)LoadImage(hInstance,MAKEINTRESOURCE(IDI\_MY50),

IMAGE\_ICON,48,48,LR\_DEFAULTCOLOR);

wcex.hCursor = LoadCursor(NULL, IDC\_ARROW);

wcex.hbrBackground = (HBRUSH)(COLOR\_WINDOW+1);

wcex.lpszMenuName = MAKEINTRESOURCE(IDC\_MY50);

wcex.lpszClassName = szWindowClass;

wcex.hIconSm = (HICON)LoadImage(hInstance,MAKEINTRESOURCE(IDI\_MY50),

IMAGE\_ICON,16,16,LR\_DEFAULTCOLOR);

return RegisterClassEx(&wcex);

}

//

// FUNCTION: InitInstance(HINSTANCE, int)

//

// PURPOSE: Saves instance handle and creates main window

//

// COMMENTS:

//

// In this function, we save the instance handle in a global variable and

// create and display the main program window.

//

BOOL InitInstance(HINSTANCE hInstance, int nCmdShow)

{

HWND hWnd;

hInst = hInstance; // Store instance handle in our global variable

hWnd = CreateWindow(szWindowClass, szTitle, WS\_CAPTION | WS\_OVERLAPPED | WS\_MINIMIZEBOX | WS\_SYSMENU,

CW\_USEDEFAULT,0, 416, 512, NULL, NULL, hInstance, NULL);

if (!hWnd)

{

return FALSE;

}

ShowWindow(hWnd, nCmdShow);

UpdateWindow(hWnd);

return TRUE;

}

void create\_bubbles(HWND hWnd)

{

int x,y;

srand ((unsigned)time(NULL));

if(!START)

for(int i=0;i<5;i++){

do {

x=rand()%9;

y=rand()%9;}

while (walkability[x][y]);

walkability[x][y] = (rand()%7)\*22+1;

COUNT\_BUBBLES++;

}

else

for(int i=0;i<3;i++)

walkability[fetus[i].x][fetus[i].y]&=zero;

for(int i=0;i<3;i++){

do {

x=rand()%9;

y=rand()%9;}

while (walkability[x][y]);

fetus[i].x=x; fetus[i].y=y;

walkability[x][y]=walkability\_fetus;

walkability[x][y]+=(rand()%7)\*22+1;

COUNT\_BUBBLES++;

if(COUNT\_BUBBLES==mapWidth\*mapHeight)

{

COUNT\_BUBBLES=0;

PlaySound (MY\_SOUND3,NULL, SND\_FILENAME | SND\_ASYNC);

START=FALSE;

if(MessageBox(NULL,\_T("GAME OVER"),\_T(""),MB\_OK))

PostMessage(hWnd,WM\_COMMAND,ID\_GAME\_NEW,0);

break;

}

else

PlaySound (MY\_SOUND2,NULL, SND\_FILENAME | SND\_ASYNC);

}

}

//

// FUNCTION: WndProc(HWND, UINT, WPARAM, LPARAM)

//

// PURPOSE: Processes messages for the main window.

//

// WM\_COMMAND - process the application menu

// WM\_PAINT - Paint the main window

// WM\_DESTROY - post a quit message and return

//

//

LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam)

{

int wmId, wmEvent;

PAINTSTRUCT ps;

static HANDLE hFile=NULL;

static int x,y;

static BITMAP bm;

static bool ANIMATE\_RUN=0;

DWORD NumberOfBytesRead;

static HDC hdc;

static HDC hdcMem;

static HBITMAP hBmpFrame;

static int COUNT\_RUN=0;

static int COUNT\_FETUS=0;

static int COUNT\_DESTROY=0;

static int LENGTH\_DESTROY=0;

static POINT \*PatchDestroy;

switch (message)

{

case WM\_LBUTTONDOWN:

if (HIWORD(lParam)<=56||COUNT\_RUN||COUNT\_FETUS||COUNT\_DESTROY||!START)

break;

if (ANIMATE\_RUN)

walkability[x][y]&=zero;

startX=x; startY=y;

x=(LOWORD(lParam)-5)/45;

y=(HIWORD(lParam)-56)/45;

if (HIWORD(walkability[x][y])^HIWORD(walkability\_fetus) && walkability[x][y])

{

ANIMATE\_RUN=TRUE;

PlaySound (MY\_SOUND0,NULL, SND\_FILENAME | SND\_ASYNC);

}

else

if (ANIMATE\_RUN)

{

ANIMATE\_RUN=FALSE;

path=FindPath (startX,startY, x, y);

if (path == found)

COUNT\_RUN=pathLength;

}

break;

case WM\_COMMAND:

wmId = LOWORD(wParam);

wmEvent = HIWORD(wParam);

// Parse the menu selections:

switch (wmId)

{

case IDM\_ABOUT:

DialogBox(hInst, MAKEINTRESOURCE(IDD\_ABOUTBOX), hWnd, About);

break;

case IDM\_EXIT:

DestroyWindow(hWnd);

break;

case ID\_FILE\_32771: //save

walkability[x][y]&=zero;

hFile=CreateFile(FILE\_PATH,GENERIC\_WRITE,FILE\_SHARE\_READ,NULL,CREATE\_ALWAYS,

FILE\_ATTRIBUTE\_NORMAL,NULL);

if(hFile)

{

WriteFile(hFile,walkability,sizeof(walkability),&NumberOfBytesRead,NULL);

WriteFile(hFile,&SCORE,sizeof(int),&NumberOfBytesRead,NULL);

WriteFile(hFile,&my\_time,sizeof(TIME),&NumberOfBytesRead,NULL);

CloseHandle(hFile);

}

else

MessageBox(hWnd,\_T("Невозможно открыть файл"),\_T("Ошибка"), MB\_ICONWARNING | MB\_OK);

break;

case ID\_FILE\_32772://load

ANIMATE\_RUN=0;

hFile=CreateFile(FILE\_PATH,GENERIC\_READ,FILE\_SHARE\_READ,NULL,OPEN\_EXISTING,

FILE\_ATTRIBUTE\_NORMAL,NULL);

if(hFile)

{

ReadFile(hFile,walkability,sizeof(walkability),&NumberOfBytesRead,NULL);

ReadFile(hFile,&SCORE,sizeof(int),&NumberOfBytesRead,NULL);

ReadFile(hFile,&my\_time,sizeof(TIME),&NumberOfBytesRead,NULL);

CloseHandle(hFile);

COUNT\_BUBBLES=0;

int k=0;

for(int i=0;i<9;i++)

for(int j=0;j<9;j++)

if( walkability[i][j])

{

COUNT\_BUBBLES++;

if(!(HIWORD(walkability[i][j])^HIWORD(walkability\_fetus)))

{

fetus[k].x=i;

fetus[k++].y=j;

}

}

START=TRUE;

T1=GetTickCount()-(my\_time.hour\*3600+my\_time.min\*60+my\_time.sec)\*1000;

}

else

MessageBox(hWnd,\_T("Невозможно открыть файл"),\_T("Ошибка"), MB\_ICONWARNING | MB\_OK);

break;

case ID\_GAME\_NEW://new

ANIMATE\_RUN=0;

for(int i=0;i<9;i++)

for(int j=0;j<9;j++)

walkability[i][j]=0;

COUNT\_BUBBLES=0;

SCORE=0;

create\_bubbles(hWnd);

START=TRUE;

T1=GetTickCount();

break;

default:

return DefWindowProc(hWnd, message, wParam, lParam);

}

break;

case WM\_CREATE:

himl=ImageList\_Create(40,40,ILC\_COLOR24,154,0);

himl2=ImageList\_Create(7,13,ILC\_COLOR32 | ILC\_MASK ,10,0);

himl3=ImageList\_Create(25,25,ILC\_COLOR24 | ILC\_MASK,7,0);

himl4=ImageList\_Create(18,35,ILC\_COLOR32 |ILC\_MASK,10,0);

ImageList\_Add(himl, hBitmapNum1, (HBITMAP)NULL);

ImageList\_Add(himl, hBitmapNum2, (HBITMAP)NULL);

ImageList\_Add(himl, hBitmapNum3, (HBITMAP)NULL);

ImageList\_Add(himl, hBitmapNum4, (HBITMAP)NULL);

ImageList\_Add(himl, hBitmapNum5, (HBITMAP)NULL);

ImageList\_Add(himl, hBitmapNum6, (HBITMAP)NULL);

ImageList\_Add(himl, hBitmapNum7, (HBITMAP)NULL);

ImageList\_Add(himl2, hBitmapNum8, hBitmapNum12);

ImageList\_Add(himl3, hBitmapNum9, hBitmapNum13);

ImageList\_Add(himl4, hBitmapNum10, hBitmapNum11);

SetTimer(hWnd,TEMER\_TIME,100,NULL);

hdc=GetDC(hWnd);

hdcMem=CreateCompatibleDC (hdc);

GetObject (hBitmapNum0, sizeof(bm), &bm);

hBmpFrame=CreateCompatibleBitmap(hdc,bm.bmWidth,bm.bmHeight);

SelectObject (hdcMem, (HBITMAP)hBmpFrame);

break;

case WM\_ERASEBKGND:

break;

case WM\_TIMER:

switch(wParam)

{

case TEMER\_TIME:

if(START)

{

T2=GetTickCount();

my\_time.hour=(T2-T1)/(1000\*3600);

my\_time.min=(T2-T1)/(1000\*60)-my\_time.hour\*60;

my\_time.sec=(T2-T1)/1000-my\_time.hour\*3600-my\_time.min\*60;

}

if(ANIMATE\_RUN)

walkability[x][y]=((((walkability[x][y] >>16)+1)%7)<<16)+(walkability[x][y]&0x0000ffff);

if(COUNT\_FETUS)

{

for (int i=0;i<3;i++)

{

if(!(HIWORD(walkability[fetus[i].x][fetus[i].y])))

{

do {

x=rand()%9;

y=rand()%9;}

while (walkability[x][y]);

walkability[x][y]=walkability\_fetus;

walkability[x][y]+=(rand()%7)\*22+1;

fetus[i].x=x; fetus[i].y=y;

}

walkability[fetus[i].x][fetus[i].y]=((((walkability[fetus[i].x][fetus[i].y] >>16)-1))<<16)+

(walkability[fetus[i].x][fetus[i].y]&zero);

}

COUNT\_FETUS--;

if(!COUNT\_FETUS)

create\_bubbles(hWnd);

}

if(COUNT\_DESTROY)

{

for(int i=0;i<LENGTH\_DESTROY;i++)

walkability[PatchDestroy[i].x][PatchDestroy[i].y]=((17-COUNT\_DESTROY)<<16)+

(walkability[PatchDestroy[i].x][PatchDestroy[i].y]&0x0000ffff);

COUNT\_DESTROY--;

if(!COUNT\_DESTROY)

{

for(int i=0;i<LENGTH\_DESTROY;i++)

walkability[PatchDestroy[i].x][PatchDestroy[i].y]=0;

dest\_len(PatchDestroy);

SCORE+=LENGTH\_DESTROY;

}

}

if (COUNT\_RUN)

{

if(COUNT\_RUN==pathLength)

{

if(!walkability[pathBank[0].x][pathBank[0].y] ||

!(HIWORD(walkability[x][y])^HIWORD(walkability\_fetus)))

{

walkability[pathBank[0].x][pathBank[0].y]=

walkability[startX][startY]&zero;

}

else

{

walkability[pathBank[1].x][pathBank[1].y]=

walkability[startX][startY]&zero;

COUNT\_RUN--;

}

walkability[startX][startY]=0;

}

else

{

if(!walkability[pathBank[pathLength-COUNT\_RUN].x][pathBank[pathLength-COUNT\_RUN].y] ||

!(HIWORD(walkability[x][y])^HIWORD(walkability\_fetus)))

{

walkability[pathBank[pathLength-COUNT\_RUN].x][pathBank[pathLength-COUNT\_RUN].y]=

walkability[pathBank[pathLength-COUNT\_RUN-1].x][pathBank[pathLength-COUNT\_RUN-1].y];

walkability[pathBank[pathLength-COUNT\_RUN-1].x][pathBank[pathLength-COUNT\_RUN-1].y]=0;

}

else

{

walkability[pathBank[pathLength-COUNT\_RUN+1].x][pathBank[pathLength-COUNT\_RUN+1].y]=

walkability[pathBank[pathLength-COUNT\_RUN-1].x][pathBank[pathLength-COUNT\_RUN-1].y];

walkability[pathBank[pathLength-COUNT\_RUN-1].x][pathBank[pathLength-COUNT\_RUN-1].y]=0;

COUNT\_RUN--;

}

}

COUNT\_RUN--;

if(COUNT\_RUN<=0)

{

if(PatchDestroy=len(x,y,LENGTH\_DESTROY))

{

COUNT\_DESTROY=10;

COUNT\_BUBBLES-=LENGTH\_DESTROY;

PlaySound (MY\_SOUND1,NULL, SND\_FILENAME | SND\_ASYNC);

}

COUNT\_FETUS=4;

path=0;

EndPathfinder();

}

}

break;

}

Paint(hdc,hdcMem);

BitBlt (hdc,0,0,bm.bmWidth,bm.bmHeight,hdcMem,0,0,SRCCOPY);

break;

case WM\_DESTROY:

ImageList\_Destroy(himl);

ImageList\_Destroy(himl2);

ImageList\_Destroy(himl3);

ImageList\_Destroy(himl4);

ReleaseDC(hWnd,hdc);

DeleteDC(hdcMem);

KillTimer(hWnd,TEMER\_TIME);

PostQuitMessage(0);

break;

default:

return DefWindowProc(hWnd, message, wParam, lParam);

}

return 0;

}

// Message handler for about box.

INT\_PTR CALLBACK About(HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)

{

UNREFERENCED\_PARAMETER(lParam);

switch (message)

{

case WM\_INITDIALOG:

return (INT\_PTR)TRUE;

case WM\_COMMAND:

if (LOWORD(wParam) == IDOK || LOWORD(wParam) == IDCANCEL)

{

EndDialog(hDlg, LOWORD(wParam));

return (INT\_PTR)TRUE;

}

break;

}

return (INT\_PTR)FALSE;

}