ТУ София ф.Пловдив

Курсов проект

Компютерна графика

Розработка на 2D компютерна игра «Тетрис» сьс исползване на GDI+

Виктор Тупиков

373575

43а

За основа взял собствен проект «[тетрис на ардуино](https://www.tinkercad.com/things/51vP7AlGOg3-tetris)» и последваща доработка за конзола windows. Модул “tetr.h” приема команди за подготвене на масиви, избиране на тип на игра, установка на сложността (функции setup\_X()), управление по време на игра (променлива scan вьв loop(typ\_gry, scan)).

scan == 1, 2, 3, 4 или 0.

1 – 4 сьответства на натиснати копчета UP, LEFT, DOWN, RIGHT,

0 – движение надолу по сработване на таймер IDT\_TIMER1 или IDT\_TIMER3.

IDT\_TIMER1 = 1сек сьс бавно намаление по гиперболическа крива (намаляне на IDT\_TIMER1 управлява от IDT\_TIMER2, коят се увеличава линейно)

IDT\_TIMER3 сьс тайминг ‘0’ задейства след натискане на DOWN, вдигна флаг fl\_s и работи до !prov\_krok() – невозможност за продолжение на движение надолу. Флаг fl\_s блокира вьвеждане от клавиатура. Идея сьс исползване на таймер сьстои в желание да пропускам чрез WinProc случайни натискания на клавиатура по време на бьрз ход надолу, а не натрупване на сьобщения, по сьща логика работи таймер IDT\_TIMER4 (задейства по време на изтриване на заполнени линии и цяло поле в края на игра (функция shmarkli()))

Взаимодействие между фигура и кошон разщитва спомощта на 2 служебни масиви mas\_rab[V\_LIN + 3][H\_LIN + 3] и fig\_tek[4][4] и координати на фигура {y,x} (‘+3’ за недопускане на излизане извьн масива при наложении mas\_rab + fig\_tek и детектиране на препятствие). функция prnt\_podg() заполнява масив mas\_pr[V\_LIN][H\_LIN].

Функция kubik() получава размери на базов блок. Макроси ‘int X/Y(float)’ позволяват да представя плот за рисуване като координатно поле X \* Y вне зависимост от размера на прозорец.

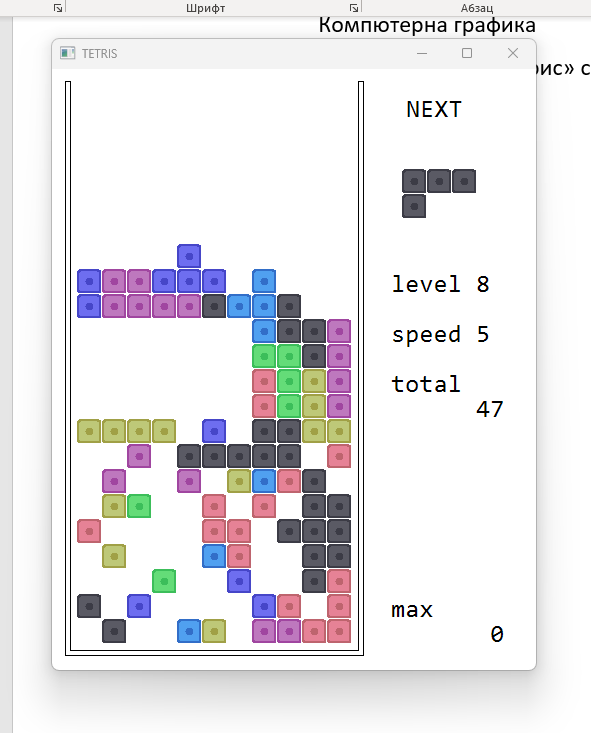
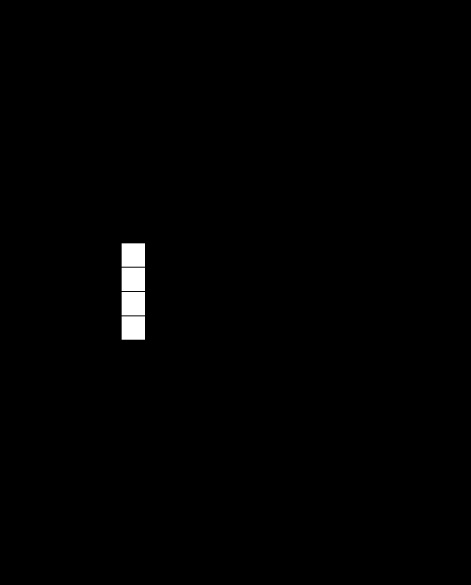
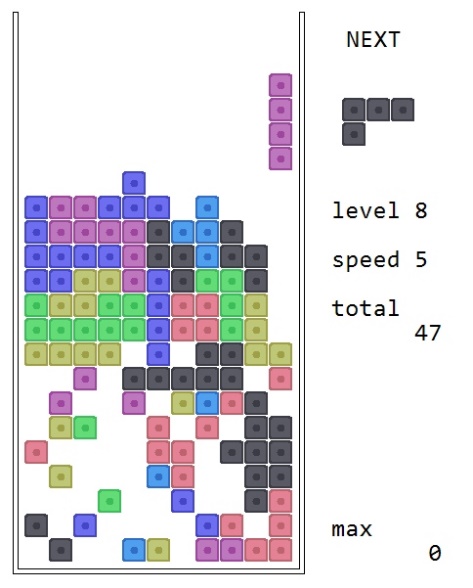
Функция рисува кош, десни титри, базов блок(kb1, kb2, kb3) сьс габарити X(1)хY(1). Вьв вложени цикли матрица премества по игрово поле и ако ‘mas\_pr[j][i] != 0’ рисува базов блок сьс сьответветния цвят. По сьщ начин рисува следваща фигура (fig\_next[4][4]). В меню за избора на тип на игра (etap == 0) вместо mas\_pr исползва статичен mas\_preview.

Накрая в зависимост от etap и pause вьрху игрово поле рисува кнопки("Normal Game", "Adventure", "LEVEL" ’↑’ ‘↓’) или сьобщения(“PAUSE”, “READY”).

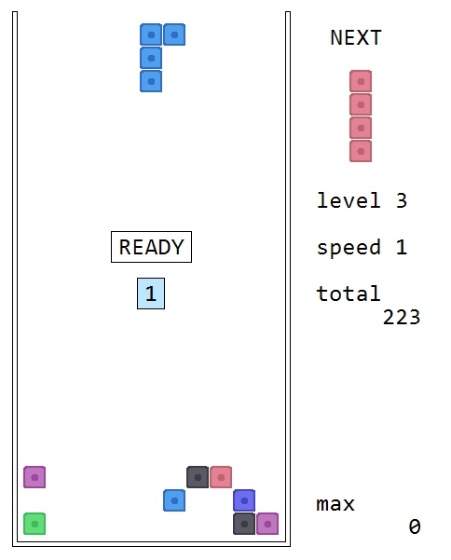
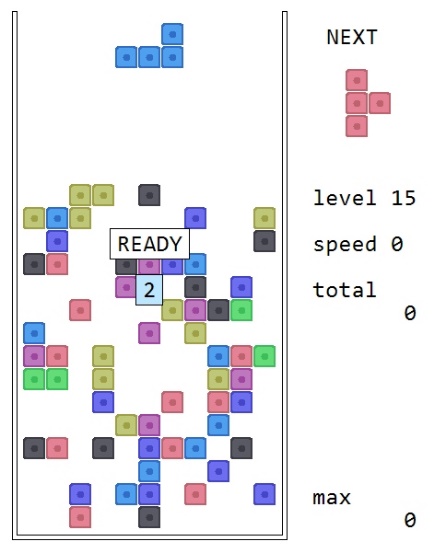
Функция GetEncoderClsid() и блок ‘if(save)’ функции DRAW\_TEST\_\_() сохраняват скриншот на отрисован вьв DRAW\_TEST() кадр (картинки в тази документ направила сама игра.)(WM\_KEYUP-> VK\_SNAPSHOT->{save = 1;updateWindow();})

Функция shmarkli() изтрива линии (soply[i] == 1(заплнява се вьв функция prov\_lin())) тьй като рисува бели квадратчета вьрху необновен кадр (без ‘graphics.Clear(Color(255, 255, 255))’ <=> ‘graphics.Clear(Color(0, 0, 0, 0))’).

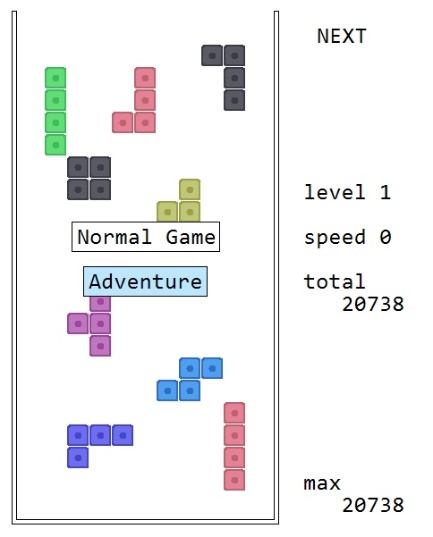
Негов таймер (IDT\_TIMER4) включва при f\_sop == 1(функция prov\_lin()) или f\_sop\_ver == 1(loop()->край на игра), блокира клавиатура и работи до f\_sop == H\_LIN



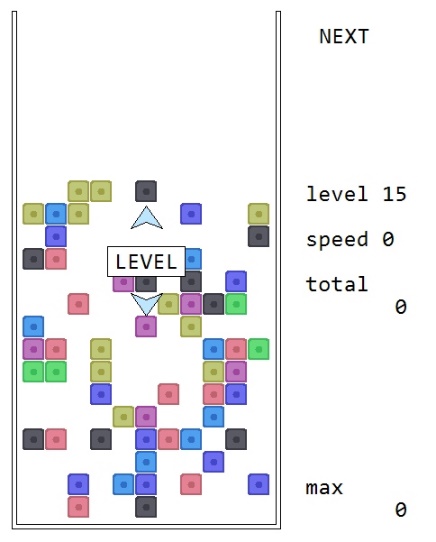
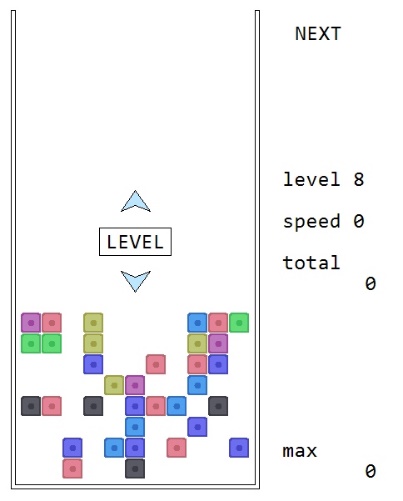
IDT\_TIMER5 отчитва секунди на (etap == 3) титр “READY” ‘3’ – ‘2’ – ‘1’; etap = 4;



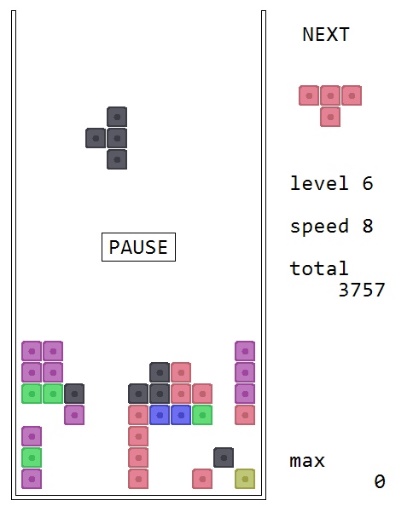
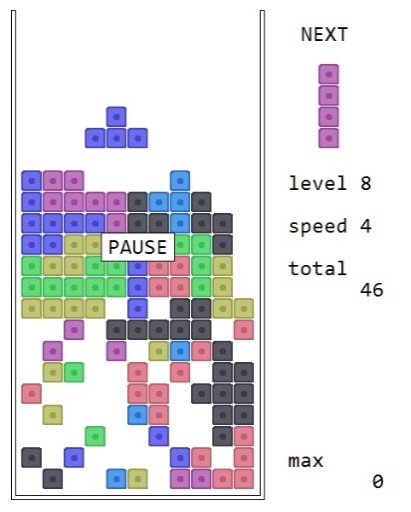
Меню при etap == 0; избор на тип на игра. Фон – mas\_ preview, ‘бутони’ изчерчени сьс gdi+, навигация сьс стрелки.



etap == 2; установка на сложност при ‘Normal game’



etap == 4 && pause == 1



Управление:

VK\_UP – ротация,

VK\_LEFT, VK\_RIGHT – ляво, дясно,

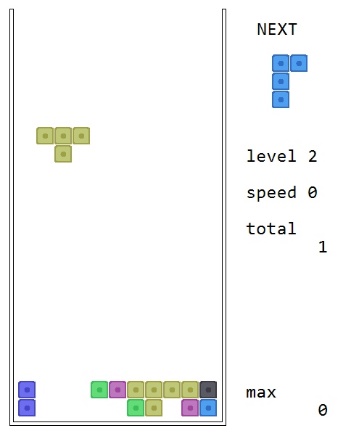
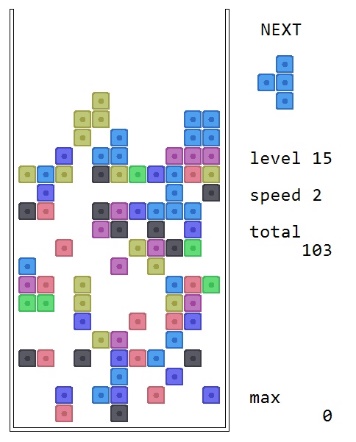
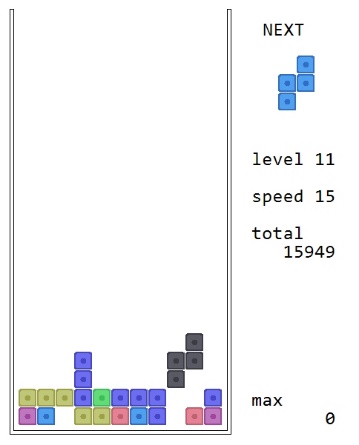
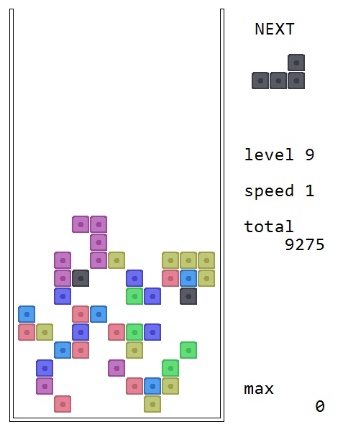
VK\_DOWN – ускорено движение надолу,

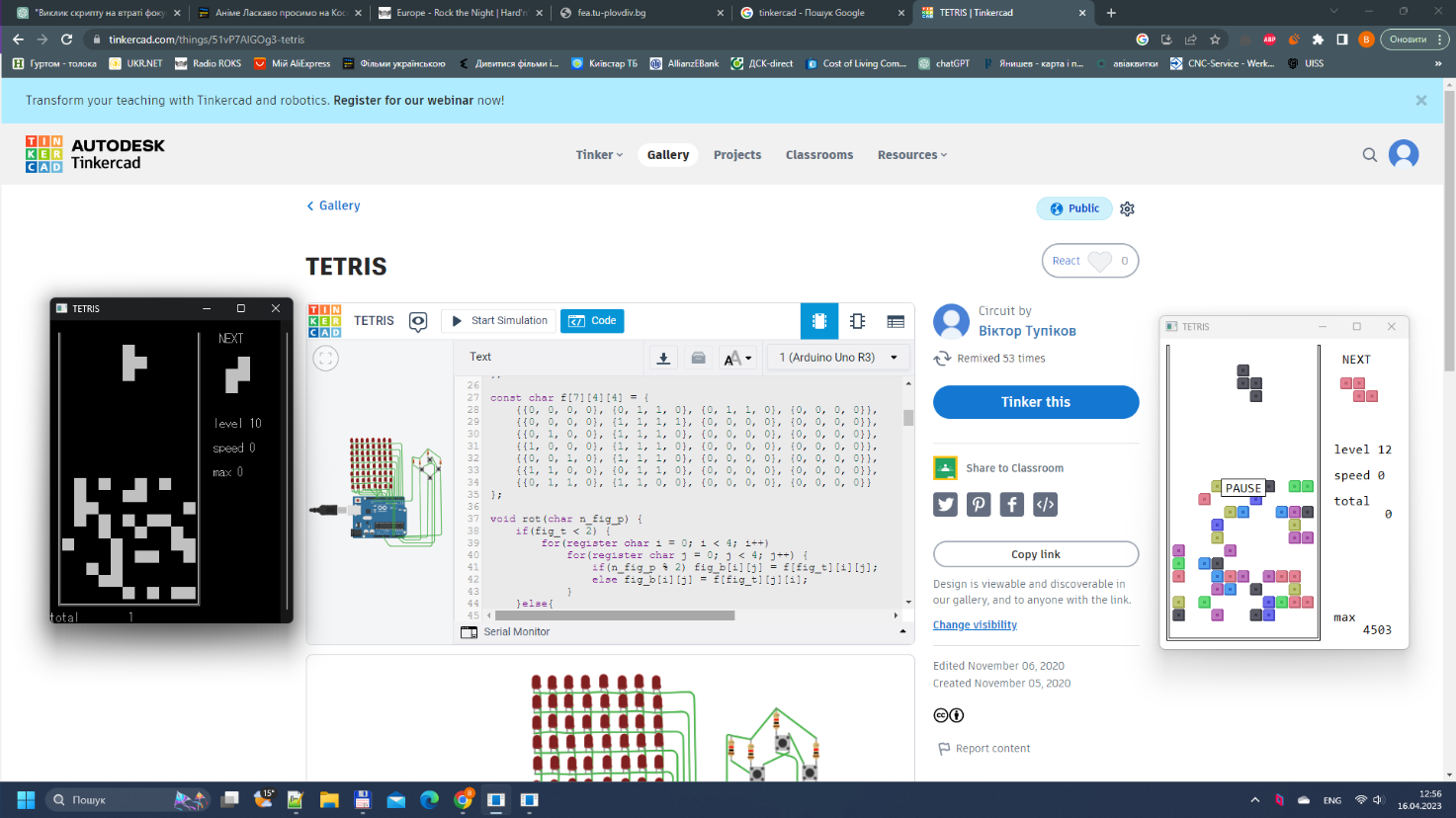
VK\_PAUSE || VK\_PRIOR || VK\_NEXT – пауза,

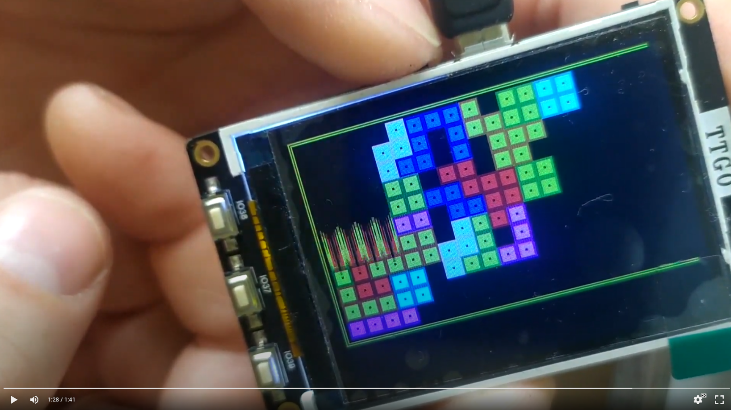
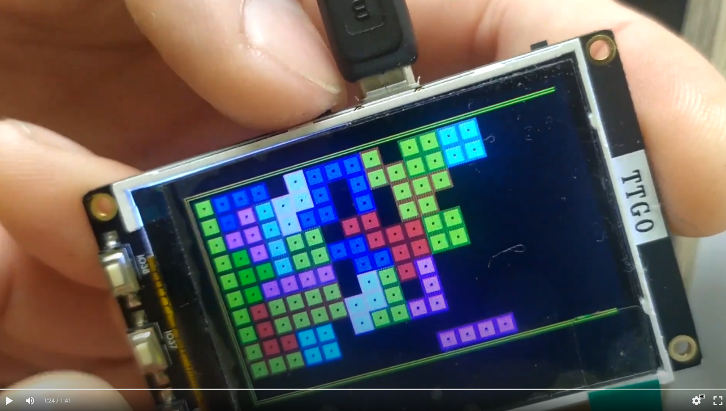
VK\_RETURN – ввод в меню etap 0 и 2,

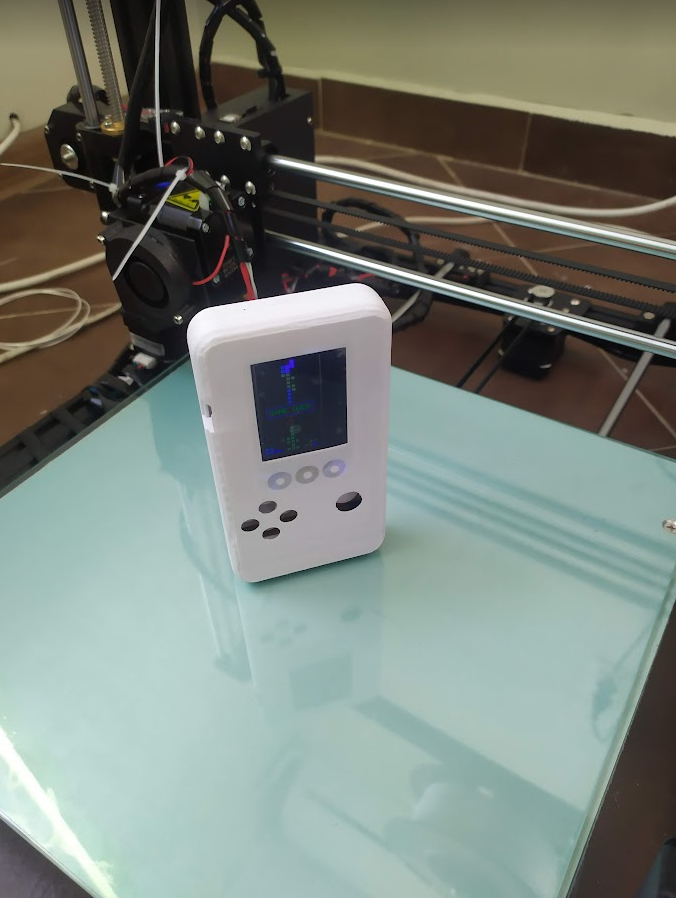
VK\_SNAPSHOT – скриншот,

VK\_ESCAPE - DestroyWindow(hhhwww);





Сьщ ардуино-тетрис, контроллер esp32 + дисплей(исп. TFT\_eSPI.h)



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* stdfx.h \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#pragma once

#include "targetver.h"

#include <windows.h>

#include <stdlib.h>

#include <malloc.h>

#include <memory.h>

#include <tchar.h>

#include <GDIPlus.h>

using namespace Gdiplus;

#include "tetr.h"

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Win32Project3.h \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#pragma once

#include "resource.h"

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Resourse.h \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#define IDS\_APP\_TITLE 103

#define IDR\_MAINFRAME 128

#define IDD\_WIN32PROJECT3\_DIALOG 102

#define IDD\_ABOUTBOX 103

#define IDM\_ABOUT 104

#define IDM\_EXIT 105

#define IDI\_WIN32PROJECT3 107

#define IDI\_SMALL 108

#define IDC\_WIN32PROJECT3 109

#define IDC\_MYICON 2

#ifndef IDC\_STATIC

#define IDC\_STATIC -1

#endif

#ifdef APSTUDIO\_INVOKED

#ifndef APSTUDIO\_READONLY\_SYMBOLS

#define \_APS\_NO\_MFC 130

#define \_APS\_NEXT\_RESOURCE\_VALUE 129

#define \_APS\_NEXT\_COMMAND\_VALUE 32771

#define \_APS\_NEXT\_CONTROL\_VALUE 1000

#define \_APS\_NEXT\_SYMED\_VALUE 110

#endif

#endif

//#define COLIR\_FONU RGB(255, 255, 255)

//#define YA(a) ((int)((a) \* yend) + ycentr)

//#define XA(a) ((int)((a) \* xend) + xcentr)

//#define YC(a) ((int)(((a) + 0.5) \* yend) + ycentr)

//#define XC(a) ((int)(((a) + 0.5) \* xend) + xcentr)

#define Y(a) ((int)((a) \* yend))

#define X(a) ((int)((a) \* xend))

#define draw\_txt(txt, x, y) graphics.DrawString(txt, -1, &font, PointF((Gdiplus::REAL)X((x)), (Gdiplus::REAL)Y(y)), &pen\_txt);

#define WM\_USER\_SHVYDKO\_VNYZ 0x8001

#define WM\_USER\_KROK\_UNYZ 0x8002

#define IDT\_TIMER1 0x8888

#define IDT\_TIMER2 0x8887

#define IDT\_TIMER3 0x8886

#define IDT\_TIMER4 0x8885

#define IDT\_TIMER5 0x8884

#define UOI\_TIMERPROC\_EXCEPTION\_SUPPRESSION 7

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* targetver.h \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#pragma once

#include <SDKDDKVer.h>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* tetr.h \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#define V\_LIN 22

#define H\_LIN 11

#define pryskorennia 200

#define shvydkist 1000

extern int mas\_pr[V\_LIN][H\_LIN];

extern int mas\_preview[V\_LIN][H\_LIN];

extern int fig\_next[4][4];

extern int fl\_s;

extern int hc;

extern int vaj\_mem\_2;

extern int total;

extern int max\_total\_norm, max\_total\_adv;

extern int pause;

extern int etap;

extern unsigned long takt, prsc;

extern int sopli[V\_LIN];

extern int f\_sop, f\_sop\_ver;

extern int ready;

extern int f\_next\_level;

int loop(int, int);

void setup\_00();

void setup\_0(int);

void setup\_1(int);

void setup\_2(int);

void setup\_3();

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* stdfx.cpp \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#include "stdafx.h"

#pragma comment(lib, "GdiPlus.lib")

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* tetr.cpp \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//#include <stdio.h>

//#include <conio.h>

//#include <Windows.h>

#include <time.h>

#include <stdlib.h>

//#include <malloc.h>

//#include <stdlib.h>

#include "tetr.h"

void rot(int n\_fig\_p);

void gen();

//void setup\_1(int);

//void setup\_2(int);

//void setup\_3();

int prov\_lin();

int prov\_krok(int new\_x, int new\_y);

int prov\_rot(int new\_fig\_p);

void prnt\_podg(int ff);

//int loop(int, int);

void gen\_vaj();

int mas\_rab[V\_LIN + 3][H\_LIN + 3];

int fig\_tec[4][4] = {0};

int fig\_b[4][4] = {0};

int fig\_next[4][4] = {0};

int x, y;

int fig\_t = 0, fig\_p = 0, fig\_t\_n = 0, fig\_p\_n = 0, fig\_c = 0, fig\_c\_n = 0;

unsigned long takt, prsc;

int fl\_s = 0;

int hc;

int vaj;

int vaj\_mem;

int vaj\_mem\_2;

int total;

int max\_total\_norm, max\_total\_adv;

int pause;

int etap = 0;

int sopli[V\_LIN];

int f\_sop, f\_sop\_ver;

int ready;

int f\_next\_level;

int mas\_pr[V\_LIN][H\_LIN] = {0};

const int mas\_preview[V\_LIN][H\_LIN] = {

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

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

{0, 3, 0, 0, 0, 2, 0, 0, 0, 1, 0},

{0, 3, 0, 0, 0, 2, 0, 0, 0, 1, 0},

{0, 3, 0, 0, 2, 2, 0, 0, 0, 0, 0},

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

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

{0, 0, 1, 1, 0, 0, 0, 5, 0, 0, 0},

{0, 0, 0, 0, 0, 0, 5, 5, 0, 0, 0},

{0, 0, 0, 0, 0, 0, 5, 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, 7, 0, 0, 0, 0, 0, 0, 0},

{0, 0, 7, 7, 0, 0, 0, 0, 0, 0, 0},

{0, 0, 0, 7, 0, 0, 0, 0, 0, 0, 0},

{0, 0, 0, 0, 0, 0, 0, 4, 4, 0, 0},

{0, 0, 0, 0, 0, 0, 4, 4, 0, 0, 0},

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

{0, 0, 6, 6, 6, 0, 0, 0, 0, 2, 0},

{0, 0, 6, 0, 0, 0, 0, 0, 0, 2, 0},

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

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

};

const int f[7][4][4] = {

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

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

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

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

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

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

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

};

void rot(int n\_fig\_p)

{

int i, j;

if(fig\_t < 2) {

for(i = 0; i < 4; i++)

for(j = 0; j < 4; j++) {

if(n\_fig\_p % 2) fig\_b[i][j] = f[fig\_t][i][j] \* fig\_c;

else fig\_b[i][j] = f[fig\_t][j][i] \* fig\_c;

}

}else{

fig\_b[1][3] = 0;

fig\_b[3][1] = 0;

if(fig\_t < 5)

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

for(j = 0; j < 3; j++) {

if(!(n\_fig\_p % 4)) fig\_b[i][j] = f[fig\_t][i][j] \* fig\_c;

else if (n\_fig\_p % 4 == 1) fig\_b[i][j] = f[fig\_t][2 - j][i] \* fig\_c;

else if (n\_fig\_p % 4 == 2) fig\_b[i][j] = f[fig\_t][2 - i][2 - j] \* fig\_c;

else fig\_b[i][j] = f[fig\_t][j][2 - i] \* fig\_c;

}

else

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

for(j = 0; j < 3; j++) {

if(!(n\_fig\_p % 2)) fig\_b[i][j] = f[fig\_t][i][j] \* fig\_c;

else fig\_b[i][j] = f[fig\_t][2 - j][i] \* fig\_c;

}

}

}

void gen(int prnt)

{

int i, j;

x = 2;

y = (H\_LIN / 2) + 1;

fig\_t = fig\_t\_n;

fig\_p = fig\_p\_n;

fig\_c = fig\_c\_n;

fig\_t\_n = rand() % 7;

fig\_p\_n = rand() % 4;

fig\_c\_n = (rand() % 7) + 1;

if(true) {

int mem[2] = {fig\_t, fig\_c};

fig\_t = fig\_t\_n;

fig\_c = fig\_c\_n;

rot(fig\_p\_n);

for(i = 0; i < 4; i++) {

for(j = 0; j < 4; j++)

fig\_next[i][j] = fig\_b[i][j];

}

fig\_t = mem[0];

fig\_c = mem[1];

rot(fig\_p);

for(i = 0; i < 4; i++)

for(j = 0; j < 4; j++)

fig\_tec[i][j] = fig\_b[i][j];

if(fig\_tec[0][0] + fig\_tec[0][1] + fig\_tec[0][2] + fig\_tec[0][3] == 0)

x--;

}

if(prnt)

total++;

}

void gen\_vaj()

{

if(vaj\_mem\_2 > vaj\_mem) {

int j = 1;

for(; j <= vaj\_mem\_2; j++)

for(int i = 1; i <= H\_LIN; i++)

mas\_rab[V\_LIN - j + 1][i] = (rand() % 3 ? 0 : 1) \* (rand() % 7 + 1);

vaj = vaj\_mem = vaj\_mem\_2;

for(; V\_LIN - j >= 0; j++)

for(int i = 1; i <= H\_LIN; i++)

mas\_rab[V\_LIN - j + 1][i] = 0;

}

else if(vaj\_mem != vaj) {

if(vaj > vaj\_mem)

for(int i = 1; i <= H\_LIN; i++) {

mas\_rab[V\_LIN - vaj + 1][i] = (rand() % 3 ? 0 : 1) \* (rand() % 7 + 1);

}

else

for(int i = 1; i <= H\_LIN; i++) {

mas\_rab[V\_LIN - vaj][i] = 0;

}

vaj\_mem\_2 = vaj\_mem = vaj;

}

}

void setup\_00()

{

max\_total\_norm = 0;

max\_total\_adv = 0;

pause = 1;

srand((unsigned int)time(NULL));

gen(0);

fl\_s = 0;

f\_sop = 0;

f\_sop\_ver = 0;

takt = shvydkist;

etap = 0;

ready = 0;

f\_next\_level = 0;

hc = rand() % 10000;

}

void setup\_0(int no\_adv)

{

vaj\_mem\_2 = no\_adv == 1 ? vaj\_mem\_2 : 1;

etap = 1;

}

void setup\_1(int no\_adv)

{

int i, j;

vaj = 0;

vaj\_mem = 0;

if(no\_adv || vaj\_mem\_2 == 1) {

gen(0);

total = 0;

}

for(i = 0; i < (V\_LIN + 3); i++)

for(j = 0; j < (H\_LIN + 3); j++) {

if(!i || !j || i == (V\_LIN + 1) || j == (H\_LIN + 1) || i == (V\_LIN + 2) || j == (H\_LIN + 2)) mas\_rab[i][j] = 1;

else mas\_rab[i][j] = 0;

}

for(i = 0; i < 4; i++)

for(j = 0; j < 4; j++)

fig\_tec[i][j] = 0;

if(!no\_adv && vaj\_mem\_2 != 18)

vaj\_mem\_2++;

gen(0);

if(vaj\_mem\_2 > 1)

do{gen\_vaj();}while(prov\_lin());

else

gen\_vaj();

prnt\_podg(0);

fl\_s = 0;

if(no\_adv) {

gen\_vaj();

prnt\_podg(0);

etap = 2;

}else {

etap = 3;

ready = 4;

}

takt = shvydkist;

prsc = pryskorennia;

}

void setup\_2(int scan)

{

if(scan) {

if(scan == 1) {

if(vaj < V\_LIN - 4) {

vaj++;

}

}

else if(scan == 3) {

if(vaj > 0) {

vaj--;

}

}

else {

etap = 3;

ready = 4;

}

}

gen\_vaj();

prnt\_podg(0);

}

void setup\_3()

{

prnt\_podg(1);

pause = 0;

etap = 4;

}

int prov\_lin()

{

f\_sop = 0;

int brbr = 0;

int i, j, ii, jj;

int ff = 0;

for(i = 1; i <= V\_LIN; i++)

for(j = 1; j <= H\_LIN; j++) {

sopli[i - 1] = 0;

if(!mas\_rab[i][j]) break;

if(j == H\_LIN) {

total += V\_LIN - j + 1;

brbr++;

sopli[i - 1] = 1;

f\_sop = 1;

for(ii = i; ii > 1; ii--)

for(jj = 1; jj < (H\_LIN + 1); jj++) mas\_rab[ii][jj] = mas\_rab[ii - 1][jj];

for(jj = 1; jj < (H\_LIN + 1); jj++) mas\_rab[1][jj] = 0;

}

}

switch(brbr) {

case 0:

break;

case 1:

total += 10;

break;

case 2:

total += 20;

break;

case 3:

total += 40;

break;

case 4:

total += 100;

break;

default:

break;

}

for(j = 1; j <= H\_LIN; j++) {

ff = 1;

for(i = V\_LIN; i > 1; i--) {

if(ff && !mas\_rab[i][j])

ff = 0;

else if(!ff && mas\_rab[i][j])

return 0;

}

}

return 1;

}

int prov\_krok(int new\_x, int new\_y)

{

int i, j;

for(i = 0; i < 4; i++)

for(j = 0; j < 4; j++)

if(fig\_tec[i][j] && mas\_rab[new\_x + i - 1][new\_y + j - 1]) return 0;

return 1;

}

int prov\_rot(int new\_fig\_p)

{

int i, j;

int ii, jj;

int ff = 0;

rot(new\_fig\_p % 4);

for(i = 0; i < 4; i++)

for(j = 0; j < 4; j++)

if(fig\_b[i][j] && mas\_rab[x + i - 1][y + j - 1])

if(x == 1) {

x++;

if(prov\_rot(new\_fig\_p))

return 1;

else {

x--;

return 0;

}

}

else {

if(j == 0) {

for(ii = 0; ii < 4; ii++)

for(jj = 0; jj < 4; jj++)

if(fig\_b[ii][jj] && mas\_rab[x + ii - 1][y + jj])

return 0;

y++;

for(i = 0; i < 4; i++)

for(j = 0; j < 4; j++)

fig\_tec[i][j] = fig\_b[i][j];

return 1;

}

else if(fig\_t == 1 && new\_fig\_p % 2 == 1) {

for(ii = 0; ii < 4; ii++)

if(fig\_b[1][ii] && mas\_rab[x][y + ii - 2])

break;

for(jj = 0; jj < 4; jj++)

if(fig\_b[1][jj] && mas\_rab[x][y + jj - 3])

break;

if(ii == 4) {

y--;

for(i = 0; i < 4; i++)

for(j = 0; j < 4; j++)

fig\_tec[i][j] = fig\_b[i][j];

return 1;

}

else if(jj == 4) {

y -= 2;

for(i = 0; i < 4; i++)

for(j = 0; j < 4; j++)

fig\_tec[i][j] = fig\_b[i][j];

return 1;

}

else

return 0;

}

else if(j == 2) {

for(ii = 0; ii < 3; ii++)

for(jj = 0; jj < 3; jj++)

if(fig\_b[ii][jj] && mas\_rab[x + ii - 1][y + jj - 2])

return 0;

y--;

for(i = 0; i < 4; i++)

for(j = 0; j < 4; j++)

fig\_tec[i][j] = fig\_b[i][j];

return 1;

}

else

return 0;

}

for(i = 0; i < 4; i++)

for(j = 0; j < 4; j++)

fig\_tec[i][j] = fig\_b[i][j];

return 1;

}

void prnt\_podg(int ff)

{

int i, j;

for(i = 1; i < (V\_LIN + 1); i++)

for(j = 1; j < (H\_LIN + 1); j++) {

mas\_pr[i - 1][j - 1] = mas\_rab[i][j];

if(i >= x - 1 && i <= x + 2 && j >= y - 1 && j <= y + 2 && etap >=3)

mas\_pr[i - 1][j - 1] = fig\_tec[i - x + 1][j - y + 1] != 0 ? fig\_tec[i - x + 1][j - y + 1] : mas\_rab[i][j];

}

}

int loop(int no\_adv, int scan)

{

switch(scan) {

case 3:{

fl\_s = 1;

break;

}

case 2:{

if(prov\_krok(x, y - 1)) {

y--;

prnt\_podg(1);

}

break;

}

case 4:{

if(prov\_krok(x, y + 1)) {

y++;

prnt\_podg(1);

}

break;

}

case 1:{

if(prov\_rot(fig\_p + 1)) {

fig\_p = (fig\_p + 1) % 4;

prnt\_podg(1);

}

break;

}

default:{

break;

}

}

if(!scan) {

int i, j;

if(prov\_krok(x + 1, y)) {

x++;

prnt\_podg(1);

}else{

for(i = 0; i < 4; i++)

for(j = 0; j < 4; j++)

mas\_rab[x + i - 1][y + j - 1] = fig\_tec[i][j] != 0 ? fig\_tec[i][j] : mas\_rab[x + i - 1][y + j - 1];

if(prov\_lin() && !no\_adv) {

f\_sop\_ver = 1;

f\_next\_level = 1;

total += vaj\_mem\_2 \* 100;

return 1;

}

gen(1);

if(!prov\_krok(x, y)) {

prnt\_podg(1);

f\_sop\_ver = 1;

if(!no\_adv)

vaj\_mem\_2 = 1;

if((no\_adv == 1 ? max\_total\_norm : max\_total\_adv) < total)

(no\_adv == 1 ? max\_total\_norm : max\_total\_adv) = total;

etap = 0;

for(i = 0; i < (V\_LIN + 3); i++)

for(j = 0; j < (H\_LIN + 3); j++) {

if(!i || !j || i == (V\_LIN + 1) || j == (H\_LIN + 1) || i == (V\_LIN + 2) || j == (H\_LIN + 2)) mas\_rab[i][j] = 1;

else mas\_rab[i][j] = 0;

}

prnt\_podg(1);

return 0;

}

prnt\_podg(1);

if(fl\_s) {

fl\_s = 0;

}

}

}

return 1;

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Win32Project3.cpp \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#pragma comment(linker,"\"/manifestdependency:type='win32' name='Microsoft.Windows.Common-Controls' version='6.0.0.0' processorArchitecture='\*' publicKeyToken='6595b64144ccf1df' language='\*'\"")

#include "stdafx.h"

#include "Win32Project3.h"

#define MAX\_LOADSTRING 100

#pragma warning(disable : 4996)

// Global Variables:

HINSTANCE hInst; // current instance

TCHAR szTitle[MAX\_LOADSTRING] = L"TETRIS"; // The title bar text

TCHAR szWindowClass[MAX\_LOADSTRING] = L"poiuytrewq"; // the main window class name

wchar\_t window\_class\_name[] = L"HOLOVNE\_VIKNO";

int ak = 0;

HWND hhhwww;

int typ\_gry = 1;

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

ATOM MyRegisterClass(HINSTANCE hInstance);

BOOL InitInstance(HINSTANCE, int);

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

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

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

int DRAW\_TEST(Bitmap\* bmp, int ak, int xend, int yend, HWND);

WNDPROC ComboBoxDefault;

VOID Example\_SaveFile(HDC hdc);

int DRAW\_TEST\_\_(HDC hdc, int ak, int xend, int yend, HWND hWnd, BOOL save);

INT GetEncoderClsid(const WCHAR\* format, CLSID\* pClsid);

int kubik(Bitmap\* bmp, int ak, int xend, int yend);

int shmarkli(Bitmap\* bmp, int ak, int xend, int yend);

int APIENTRY \_tWinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPTSTR lpCmdLine, int nCmdShow)

{

UNREFERENCED\_PARAMETER(hPrevInstance);

UNREFERENCED\_PARAMETER(lpCmdLine);

MSG msg;

HACCEL hAccelTable;

GdiplusStartupInput gdiplusStartupInput;//GdiplusStartupInput GdiplusStartupInput

ULONG\_PTR gdiplusToken;

GdiplusStartup(&gdiplusToken, &gdiplusStartupInput, NULL);

setup\_00();

MyRegisterClass(hInstance);

if(!InitInstance (hInstance, nCmdShow)) {

return FALSE;

}

hAccelTable = LoadAccelerators(hInstance, MAKEINTRESOURCE(IDC\_WIN32PROJECT3));

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

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

TranslateMessage(&msg);

DispatchMessage(&msg);

}

}

GdiplusShutdown(gdiplusToken);

KillTimer(hhhwww, IDT\_TIMER1);

KillTimer(hhhwww, IDT\_TIMER2);

KillTimer(hhhwww, IDT\_TIMER3);

KillTimer(hhhwww, IDT\_TIMER4);

KillTimer(hhhwww, IDT\_TIMER5);

return (int) msg.wParam;

}

ATOM MyRegisterClass(HINSTANCE hInstance)

{

WNDCLASSEX wincl;

wincl.hInstance = hInstance;

wincl.lpszClassName = window\_class\_name;

wincl.lpfnWndProc = WndProc;

wincl.style = CS\_DBLCLKS | CS\_HREDRAW | CS\_VREDRAW;

wincl.cbSize = sizeof(WNDCLASSEX);

wincl.hIcon = LoadIcon(hInstance, MAKEINTRESOURCE(IDI\_WIN32PROJECT3));

wincl.hIconSm = LoadIcon(hInstance, MAKEINTRESOURCE(IDI\_SMALL));

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

wincl.lpszMenuName = NULL;

wincl.cbClsExtra = 0;

wincl.cbWndExtra = 0;

wincl.hbrBackground = (HBRUSH)CreateSolidBrush(COLIR\_FONU); /\* Створюємо об'єкт, який буде заповняти кольором полотно вікна \*/

return RegisterClassEx(&wincl);

}

BOOL InitInstance(HINSTANCE hInstance, int nCmdShow)

{

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

hhhwww = CreateWindowEx(0, window\_class\_name, szTitle, WS\_OVERLAPPEDWINDOW, CW\_USEDEFAULT, 0, CW\_USEDEFAULT, 0, NULL, NULL, hInstance, NULL);

SetWindowPos(hhhwww, NULL, 0, 0, 500, 640, SWP\_NOMOVE);

SetUserObjectInformation(GetCurrentProcess(), UOI\_TIMERPROC\_EXCEPTION\_SUPPRESSION, false, 1);

SetTimer(hhhwww, IDT\_TIMER1, shvydkist, (TIMERPROC)NULL);

SetTimer(hhhwww, IDT\_TIMER2, pryskorennia, (TIMERPROC)NULL);

SetTimer(hhhwww, IDT\_TIMER3, 200000, (TIMERPROC)NULL);

SetTimer(hhhwww, IDT\_TIMER4, 200000, (TIMERPROC)NULL);

SetTimer(hhhwww, IDT\_TIMER5, 200000, (TIMERPROC)NULL);

ShowWindow(hhhwww, nCmdShow);

UpdateWindow(hhhwww);

return TRUE;

}

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

{

if(f\_sop == 1 || f\_sop\_ver == 1)

SetTimer(hhhwww, IDT\_TIMER4, 0, (TIMERPROC)NULL);

if(ready == 4)

SetTimer(hhhwww, IDT\_TIMER5, 1000, (TIMERPROC)NULL);

int wmId, wmEvent;

PAINTSTRUCT ps;

HDC hdc;

static BOOL save = 0;

switch (message) {

case WM\_TIMER: {

if(wParam == IDT\_TIMER1 && etap == 4 && pause == 0) {

SendMessage(hhhwww, (UINT)WM\_USER\_KROK\_UNYZ, (WPARAM)0, (LPARAM)0);

}else

if(wParam == IDT\_TIMER2 && etap == 4 && pause == 0) {

if (takt > 40) {

takt--;

prsc++;

SetTimer(hhhwww, IDT\_TIMER2, prsc, (TIMERPROC)NULL);

}

else {

takt = shvydkist;

prsc = pryskorennia;

SetTimer(hhhwww, IDT\_TIMER1, takt, (TIMERPROC)NULL);

SetTimer(hhhwww, IDT\_TIMER2, prsc, (TIMERPROC)NULL);

}

}else

if(wParam == IDT\_TIMER3 && etap == 4 && pause == 0) {

SendMessage(hhhwww, (UINT)WM\_USER\_SHVYDKO\_VNYZ, (WPARAM)0, (LPARAM)0);

}else

if(wParam == IDT\_TIMER4 && (f\_sop || f\_sop\_ver) && pause == 0 && ready == 0) {

InvalidateRect(hWnd,NULL, false);

UpdateWindow(hWnd);

}else

if(wParam == IDT\_TIMER5 && etap == 3) {

InvalidateRect(hWnd,NULL, false);

UpdateWindow(hWnd);

}

return 0;

}

case WM\_USER\_SHVYDKO\_VNYZ: {

if(!fl\_s) {

SetTimer(hhhwww, IDT\_TIMER3, 200000, (TIMERPROC)NULL);

SetTimer(hhhwww, IDT\_TIMER1, takt, (TIMERPROC)NULL);

return 0;

}

SetTimer(hhhwww, IDT\_TIMER3, 0, (TIMERPROC)NULL);

if(!f\_sop && !f\_sop\_ver)

loop(typ\_gry, 0);

InvalidateRect(hWnd,NULL, false);

UpdateWindow(hWnd);

return 0;

}

case WM\_USER\_KROK\_UNYZ: {

if(!f\_sop && !f\_sop\_ver)

loop(typ\_gry, 0);

InvalidateRect(hWnd,NULL, false);

UpdateWindow(hWnd);

return 0;

}

case WM\_KEYUP: {

if(wParam == VK\_SNAPSHOT) {

save = 1;

InvalidateRect(hWnd,NULL, false);

UpdateWindow(hWnd);

}

return 0;

}

case WM\_KEYDOWN:

if(wParam == VK\_ESCAPE) {

DestroyWindow(hhhwww);

return 0;

}

if(etap == 3)

return 0;

if(wParam == VK\_RETURN) {

if(etap == 0) {

setup\_0(typ\_gry);

setup\_1(typ\_gry);

}else if(etap == 2) {

setup\_2(5);

SetTimer(hhhwww, IDT\_TIMER1, takt, (TIMERPROC)NULL);

SetTimer(hhhwww, IDT\_TIMER2, prsc, (TIMERPROC)NULL);

}

}

if(wParam == VK\_PAUSE || wParam == VK\_PRIOR || wParam == VK\_NEXT) {

if(etap == 4) {

pause = pause == 1 ? 0 : 1;

SetTimer(hhhwww, IDT\_TIMER1, takt, (TIMERPROC)NULL);

}

}

if(wParam == VK\_DOWN && f\_sop == 0 && f\_sop\_ver == 0) {

if(etap == 4 && pause == 0 && fl\_s == 0) {

loop(typ\_gry, 3);

SendMessage(hhhwww, WM\_USER\_SHVYDKO\_VNYZ, (WPARAM)0, (LPARAM)0);

}else if(etap == 2) {

setup\_2(3);

}else if(etap == 0) {

typ\_gry = 0;

}

}

if(wParam == VK\_LEFT && etap == 4 && pause == 0 && fl\_s == 0 && f\_sop == 0 && f\_sop\_ver == 0) {

loop(typ\_gry, 2);

}

if(wParam == VK\_UP && f\_sop == 0 && f\_sop\_ver == 0) {

if(etap == 4 && pause == 0 && fl\_s == 0) {

loop(typ\_gry, 1);

}else if(etap == 2) {

setup\_2(1);

}else if(etap == 0) {

typ\_gry = 1;

}

}

if(wParam == VK\_RIGHT && etap == 4 && pause == 0 && fl\_s == 0 && f\_sop == 0 && f\_sop\_ver == 0) {

loop(typ\_gry, 4);

}

InvalidateRect(hWnd,NULL, false);

UpdateWindow(hWnd);

break;

case WM\_COMMAND:

wmId = LOWORD(wParam);

wmEvent = HIWORD(wParam);

switch (wmId) {

case IDM\_ABOUT:

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

break;

case IDM\_EXIT:

DestroyWindow(hWnd);

break;

default:

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

}

break;

case WM\_PAINT:

hdc = BeginPaint(hWnd, &ps);

RECT rt;

GetClientRect(hWnd, &rt);

ak = DRAW\_TEST\_\_(hdc, ak, int(rt.right),int(rt.bottom), hWnd, save);

save = 0;

EndPaint(hWnd, &ps);

break;

case WM\_DESTROY:

PostQuitMessage(0);

break;

default:

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

}

return 0;

}

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;

}

int kubik(Bitmap\* bmp, int ak, int xend, int yend)

{

xend = yend;

Graphics graphics(bmp); // създаване на обект от буфера

graphics.Clear(Color(255, 255, 255));

FontFamily fontFamily(L"Consolas");

Font font(&fontFamily, (Gdiplus::REAL)Y(1), FontStyleRegular, UnitPixel);

SolidBrush pen\_txt(Color(255, 0, 0, 0));

wchar\_t wbuf[19];

draw\_txt(L"NEXT", H\_LIN + 3, 1);

swprintf\_s(wbuf, 18, L"level %d", vaj\_mem\_2);

draw\_txt(wbuf, H\_LIN + 2.4, 8);

swprintf\_s(wbuf, 18, L"speed %d", (shvydkist - takt) / 40);

draw\_txt(wbuf, H\_LIN + 2.4, 10);

swprintf\_s(wbuf, 18, L"% 8d", total);

draw\_txt(L"total", H\_LIN + 2.4, 12);

draw\_txt(wbuf, H\_LIN + 2.4, 13);

swprintf\_s(wbuf, 18, L"% 8d", (typ\_gry == 1 ? max\_total\_norm : max\_total\_adv));

draw\_txt(L"max", H\_LIN + 2.4, V\_LIN - 1);

draw\_txt(wbuf, H\_LIN + 2.4, V\_LIN);

Pen Pen1(Color(255, 0, 0, 0)); // дефиниране на син молив

graphics.DrawLine(&Pen1, X(0.55), Y(0.5), X(0.55), Y(1.45 + V\_LIN));

graphics.DrawLine(&Pen1, X(0.75), Y(0.5), X(0.75), Y(1.25 + V\_LIN));

graphics.DrawLine(&Pen1, X(0.55), Y(0.5), X(0.75), Y(0.5));

graphics.DrawLine(&Pen1, X(1.25 + H\_LIN), Y(0.5), X(1.25 + H\_LIN), Y(1.25 + V\_LIN));

graphics.DrawLine(&Pen1, X(1.45 + H\_LIN), Y(0.5), X(1.45 + H\_LIN), Y(1.45 + V\_LIN));

graphics.DrawLine(&Pen1, X(1.25 + H\_LIN), Y(0.5), X(1.45 + H\_LIN), Y(0.5));

graphics.DrawLine(&Pen1, X(0.75), Y(1.25 + V\_LIN), X(1.25 + H\_LIN), Y(1.25 + V\_LIN));

graphics.DrawLine(&Pen1, X(0.55), Y(1.45 + V\_LIN), X(1.45 + H\_LIN), Y(1.45 + V\_LIN));

GraphicsPath kb1, kb2, kb3;

kb1.StartFigure();

kb1.AddArc(X(0.01), Y(0.01), X(0.18), Y(0.18), (REAL)180, (REAL)90);

kb1.AddArc(X(0.81), Y(0.01), X(0.18), Y(0.18), (REAL)270, (REAL)90);

kb1.AddArc(X(0.81), Y(0.81), X(0.18), Y(0.18), (REAL)0, (REAL)90);

kb1.AddArc(X(0.01), Y(0.81), X(0.18), Y(0.18), (REAL)90, (REAL)90);

kb1.CloseFigure();

kb2.StartFigure();

kb2.AddArc(X(0.1), Y(0.1), X(0.16), Y(0.16), (REAL)180, (REAL)90);

kb2.AddArc(X(0.74), Y(0.1), X(0.16), Y(0.16), (REAL)270, (REAL)90);

kb2.AddArc(X(0.74), Y(0.74), X(0.16), Y(0.16), (REAL)0, (REAL)90);

kb2.AddArc(X(0.1), Y(0.74), X(0.16), Y(0.16), (REAL)90, (REAL)90);

kb2.CloseFigure();

kb3.StartFigure();

kb3.AddArc(X(0.33), Y(0.33), X(0.34), Y(0.34), (REAL)0, (REAL)360);

kb3.CloseFigure();

Matrix myMatrix;

myMatrix.Translate((float)X(1), (float)Y(1));

for(int i = 0; i < H\_LIN; i++) {

for(int j = 0; j < V\_LIN; j++) {

ak = etap == 0 ? mas\_preview[j][i] : mas\_pr[j][i];

kb1.Transform(&myMatrix);

kb2.Transform(&myMatrix);

kb3.Transform(&myMatrix);

switch(ak % 8) {

case 1:{

SolidBrush farb\_2(Color(255, 60, 60, 70));

SolidBrush farb\_3(Color(255, 90, 90, 100));

graphics.FillPath(&farb\_2, &kb1);

graphics.FillPath(&farb\_3, &kb2);

graphics.FillPath(&farb\_2, &kb3);

break;

}

case 2:{

SolidBrush farb\_2(Color(255, 190, 100, 110));

SolidBrush farb\_3(Color(255, 230, 130, 150));

graphics.FillPath(&farb\_2, &kb1);

graphics.FillPath(&farb\_3, &kb2);

graphics.FillPath(&farb\_2, &kb3);

break;

}

case 3:{

SolidBrush farb\_2(Color(255, 60, 190, 90));

SolidBrush farb\_3(Color(255, 100, 220, 120));

graphics.FillPath(&farb\_2, &kb1);

graphics.FillPath(&farb\_3, &kb2);

graphics.FillPath(&farb\_2, &kb3);

break;

}

case 4:{

SolidBrush farb\_2(Color(255, 50, 110, 200));

SolidBrush farb\_3(Color(255, 80, 160, 240));

graphics.FillPath(&farb\_2, &kb1);

graphics.FillPath(&farb\_3, &kb2);

graphics.FillPath(&farb\_2, &kb3);

break;

}

case 5:{

SolidBrush farb\_2(Color(255, 160, 160, 70));

SolidBrush farb\_3(Color(255, 190, 200, 120));

graphics.FillPath(&farb\_2, &kb1);

graphics.FillPath(&farb\_3, &kb2);

graphics.FillPath(&farb\_2, &kb3);

break;

}

case 6:{

SolidBrush farb\_2(Color(255, 70, 70, 200));

SolidBrush farb\_3(Color(255, 110, 110, 240));

graphics.FillPath(&farb\_2, &kb1);

graphics.FillPath(&farb\_3, &kb2);

graphics.FillPath(&farb\_2, &kb3);

break;

}

case 7:{

SolidBrush farb\_2(Color(255, 160, 70, 160));

SolidBrush farb\_3(Color(255, 190, 120, 190));

graphics.FillPath(&farb\_2, &kb1);

graphics.FillPath(&farb\_3, &kb2);

graphics.FillPath(&farb\_2, &kb3);

break;

}

default: break;

} //1-7 - кольори, 0 - затерти

myMatrix.Reset();

myMatrix.Translate((float)X(0), (float)Y(1));

}

myMatrix.Translate((float)X(1), (float)Y(-V\_LIN));

}

myMatrix.Translate((float)X(2), (float)Y(2));

if(etap >= 3)

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

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

ak = fig\_next[j][i];

kb1.Transform(&myMatrix);

kb2.Transform(&myMatrix);

kb3.Transform(&myMatrix);

switch(ak % 8) {

case 1:{

SolidBrush farb\_2(Color(255, 60, 60, 70));

SolidBrush farb\_3(Color(255, 90, 90, 100));

graphics.FillPath(&farb\_2, &kb1);

graphics.FillPath(&farb\_3, &kb2);

graphics.FillPath(&farb\_2, &kb3);

break;

}

case 2:{

SolidBrush farb\_2(Color(255, 190, 100, 110));

SolidBrush farb\_3(Color(255, 230, 130, 150));

graphics.FillPath(&farb\_2, &kb1);

graphics.FillPath(&farb\_3, &kb2);

graphics.FillPath(&farb\_2, &kb3);

break;

}

case 3:{

SolidBrush farb\_2(Color(255, 60, 190, 90));

SolidBrush farb\_3(Color(255, 100, 220, 120));

graphics.FillPath(&farb\_2, &kb1);

graphics.FillPath(&farb\_3, &kb2);

graphics.FillPath(&farb\_2, &kb3);

break;

}

case 4:{

SolidBrush farb\_2(Color(255, 50, 110, 200));

SolidBrush farb\_3(Color(255, 80, 160, 240));

graphics.FillPath(&farb\_2, &kb1);

graphics.FillPath(&farb\_3, &kb2);

graphics.FillPath(&farb\_2, &kb3);

break;

}

case 5:{

SolidBrush farb\_2(Color(255, 160, 160, 70));

SolidBrush farb\_3(Color(255, 190, 200, 120));

graphics.FillPath(&farb\_2, &kb1);

graphics.FillPath(&farb\_3, &kb2);

graphics.FillPath(&farb\_2, &kb3);

break;

}

case 6:{

SolidBrush farb\_2(Color(255, 70, 70, 200));

SolidBrush farb\_3(Color(255, 110, 110, 240));

graphics.FillPath(&farb\_2, &kb1);

graphics.FillPath(&farb\_3, &kb2);

graphics.FillPath(&farb\_2, &kb3);

break;

}

case 7:{

SolidBrush farb\_2(Color(255, 160, 70, 160));

SolidBrush farb\_3(Color(255, 190, 120, 190));

graphics.FillPath(&farb\_2, &kb1);

graphics.FillPath(&farb\_3, &kb2);

graphics.FillPath(&farb\_2, &kb3);

break;

}

default: break;

} //1-7 - кольори, 0 - затерти

myMatrix.Reset();

myMatrix.Translate((float)X(0), (float)Y(1));

}

myMatrix.Translate((float)X(1), (float)Y(-4));

}

switch(etap) {

case 0:{

GraphicsPath kb7, kb6;

SolidBrush wite(Color(255, 255, 255, 255));

SolidBrush blue(Color(255, 188, 231, 255));

kb6.AddLine(X(9.81), Y(10 - 0.08), X(9.81), Y(10 + 1.27));

kb6.AddLine(X(3.22), Y(10 + 1.27), X(3.22), Y(10 - 0.08));

if(typ\_gry == 0)

graphics.FillPath(&wite, &kb6);

else

graphics.FillPath(&blue, &kb6);

draw\_txt(L"Normal Game", 3.3, 10);

graphics.DrawLine(&Pen1, X(9.81), Y(10 - 0.08), X(9.81), Y(10 + 1.27));

graphics.DrawLine(&Pen1, X(9.81), Y(10 + 1.27), X(3.22), Y(10 + 1.27));

graphics.DrawLine(&Pen1, X(3.22), Y(10 + 1.27), X(3.22), Y(10 - 0.08));

graphics.DrawLine(&Pen1, X(3.22), Y(10 - 0.08), X(9.81), Y(10 - 0.08));

kb7.AddLine(X(9.28), Y(12 - 0.08), X(9.28), Y(12 + 1.27));

kb7.AddLine(X(3.72), Y(12 + 1.27), X(3.72), Y(12 - 0.08));

if(typ\_gry == 1)

graphics.FillPath(&wite, &kb7);

else

graphics.FillPath(&blue, &kb7);

draw\_txt(L"Adventure", 3.8, 12);

graphics.DrawLine(&Pen1, X(9.28), Y(12 - 0.08), X(9.28), Y(12 + 1.27));

graphics.DrawLine(&Pen1, X(9.28), Y(12 + 1.27), X(3.72), Y(12 + 1.27));

graphics.DrawLine(&Pen1, X(3.72), Y(12 + 1.27), X(3.72), Y(12 - 0.08));

graphics.DrawLine(&Pen1, X(3.72), Y(12 - 0.08), X(9.28), Y(12 - 0.08));

break;

}

case 1:{

break;

}

case 2:{

GraphicsPath kb5, kb6, kb7;

SolidBrush wite(Color(255, 255, 255, 255));

SolidBrush blue(Color(255, 188, 231, 255));

kb5.AddLine(X(8.23), Y(11 - 0.08), X(8.23), Y(11 + 1.27));

kb5.AddLine(X(4.77), Y(11 + 1.27), X(4.77), Y(11 - 0.08));

graphics.FillPath(&wite, &kb5);

draw\_txt(L"LEVEL", 4.95, 11);

graphics.DrawLine(&Pen1, X(8.23), Y(11 - 0.08), X(8.23), Y(11 + 1.27));

graphics.DrawLine(&Pen1, X(8.23), Y(11 + 1.27), X(4.77), Y(11 + 1.27));

graphics.DrawLine(&Pen1, X(4.77), Y(11 + 1.27), X(4.77), Y(11 - 0.08));

graphics.DrawLine(&Pen1, X(4.77), Y(11 - 0.08), X(8.23), Y(11 - 0.08));

kb6.AddLine(X(6.5), Y(14), X(5.8), Y(13));

kb6.AddLine(X(5.8), Y(13), X(6.5), Y(13.3));

kb6.AddLine(X(6.5), Y(13.3), X(7.2), Y(13));

kb6.AddLine(X(7.2), Y(13), X(6.5), Y(14));

graphics.FillPath(&blue, &kb6);

graphics.DrawLine(&Pen1, X(6.5), Y(14), X(5.8), Y(13));

graphics.DrawLine(&Pen1, X(5.8), Y(13), X(6.5), Y(13.3));

graphics.DrawLine(&Pen1, X(6.5), Y(13.3), X(7.2), Y(13));

graphics.DrawLine(&Pen1, X(7.2), Y(13), X(6.5), Y(14));

kb7.AddLine(X(6.5), Y(9.15), X(5.8), Y(10.15));

kb7.AddLine(X(5.8), Y(10.15), X(6.5), Y(9.85));

kb7.AddLine(X(6.5), Y(9.85), X(7.2), Y(10.15));

kb7.AddLine(X(7.2), Y(10.15), X(6.5), Y(9.15));

graphics.FillPath(&blue, &kb7);

graphics.DrawLine(&Pen1, X(6.5), Y(9.15), X(5.8), Y(10.15));

graphics.DrawLine(&Pen1, X(5.8), Y(10.15), X(6.5), Y(9.85));

graphics.DrawLine(&Pen1, X(6.5), Y(9.85), X(7.2), Y(10.15));

graphics.DrawLine(&Pen1, X(7.2), Y(10.15), X(6.5), Y(9.15));

break;

}

case 3:{

GraphicsPath kb7, kb6;

SolidBrush wite(Color(255, 255, 255, 255));

SolidBrush blue(Color(255, 188, 231, 255));

kb6.AddLine(X(8.23), Y(10 - 0.08), X(8.23), Y(10 + 1.27));

kb6.AddLine(X(4.77), Y(10 + 1.27), X(4.77), Y(10 - 0.08));

graphics.FillPath(&wite, &kb6);

draw\_txt(L"READY", 4.95, 10);

graphics.DrawLine(&Pen1, X(8.23), Y(10 - 0.08), X(8.23), Y(10 + 1.27));

graphics.DrawLine(&Pen1, X(8.23), Y(10 + 1.27), X(4.77), Y(10 + 1.27));

graphics.DrawLine(&Pen1, X(4.77), Y(10 + 1.27), X(4.77), Y(10 - 0.08));

graphics.DrawLine(&Pen1, X(4.77), Y(10 - 0.08), X(8.23), Y(10 - 0.08));

kb7.AddLine(X(7.07), Y(12 - 0.08), X(7.07), Y(12 + 1.27));

kb7.AddLine(X(5.88), Y(12 + 1.27), X(5.88), Y(12 - 0.08));

graphics.FillPath(&blue, &kb7);

ready--;

swprintf\_s(wbuf, 18, L"%d", ready);

draw\_txt(wbuf, 6.06, 12);

graphics.DrawLine(&Pen1, X(7.07), Y(12 - 0.08), X(7.07), Y(12 + 1.27));

graphics.DrawLine(&Pen1, X(7.07), Y(12 + 1.27), X(5.88), Y(12 + 1.27));

graphics.DrawLine(&Pen1, X(5.88), Y(12 + 1.27), X(5.88), Y(12 - 0.08));

graphics.DrawLine(&Pen1, X(5.88), Y(12 - 0.08), X(7.07), Y(12 - 0.08));

if(ready == 0) {

setup\_3();

SetTimer(hhhwww, IDT\_TIMER5, 200000, (TIMERPROC)NULL);

SetTimer(hhhwww, IDT\_TIMER1, shvydkist, (TIMERPROC)NULL);

SetTimer(hhhwww, IDT\_TIMER2, pryskorennia, (TIMERPROC)NULL);

}

break;

}

case 4:{

if(pause == 1) {

GraphicsPath kb5;

SolidBrush wite(Color(255, 255, 255, 255));

kb5.AddLine(X(8.23), Y(11 - 0.08), X(8.23), Y(11 + 1.27));

kb5.AddLine(X(4.77), Y(11 + 1.27), X(4.77), Y(11 - 0.08));

graphics.FillPath(&wite, &kb5);

draw\_txt(L"PAUSE", 4.95, 11);

graphics.DrawLine(&Pen1, X(8.23), Y(11 - 0.08), X(8.23), Y(11 + 1.27));

graphics.DrawLine(&Pen1, X(8.23), Y(11 + 1.27), X(4.77), Y(11 + 1.27));

graphics.DrawLine(&Pen1, X(4.77), Y(11 + 1.27), X(4.77), Y(11 - 0.08));

graphics.DrawLine(&Pen1, X(4.77), Y(11 - 0.08), X(8.23), Y(11 - 0.08));

}

break;

}

default:{

break;

}

}

return 0;

}

INT GetEncoderClsid(const WCHAR\* format, CLSID\* pClsid) // helper function

{

UINT num = 0; // number of image encoders//помощна функция за извличане на параметри на енкодера

UINT size = 0; // size of the image encoder array in bytes

ImageCodecInfo\* pImageCodecInfo = NULL;

GetImageEncodersSize(&num, &size);

if(size == 0)

return -1; // Failure

pImageCodecInfo = (ImageCodecInfo\*)(malloc(size));

if(pImageCodecInfo == NULL)

return -1; // Failure

GetImageEncoders(num, size, pImageCodecInfo);

for(UINT j = 0; j < num; ++j)

{

if( wcscmp(pImageCodecInfo[j].MimeType, format) == 0 )

{

\*pClsid = pImageCodecInfo[j].Clsid;

free(pImageCodecInfo);

return j; // Success

}

}

free(pImageCodecInfo);

return -1; // Failure

}

int DRAW\_TEST\_\_(HDC hdc, int ak, int xend, int yend, HWND hWnd, BOOL save)

{

Graphics graphics(hdc);

Bitmap bmp(xend, yend, &graphics);

ak = DRAW\_TEST(&bmp, ak, xend, yend, hWnd);

graphics.DrawImage(&bmp, 0, 0);

static int im = 0;

if(save) {

save = 0;

im++;

wchar\_t wbuff[20];

CLSID clsid;

swprintf(wbuff, L"Image\_%4d\_%d.jpg", hc, im);

EncoderParameters encoderParameters;

ULONG quality;

GetEncoderClsid(L"image/jpeg",&clsid);

encoderParameters.Count = 1;

encoderParameters.Parameter[0].Guid = EncoderQuality;

encoderParameters.Parameter[0].Type = EncoderParameterValueTypeLong;

encoderParameters.Parameter[0].NumberOfValues = 1;

// запис на JPEG формат изображение с компресия 99.

quality = 99;

encoderParameters.Parameter[0].Value = &quality;

bmp.Save(wbuff, &clsid, &encoderParameters);

}

return ak;

}

int DRAW\_TEST(Bitmap\* bmp, int ak, int xend, int yend, HWND hWnd)

{

if(xend / (H\_LIN + 8) > yend / (V\_LIN + 2)) {

yend /= (V\_LIN + 2);

xend = yend;

}else {

xend /= (H\_LIN + 8);

yend = xend;

}

if(!f\_sop && !f\_sop\_ver)

kubik(bmp, 0, xend, xend);

else

shmarkli(bmp, 0, xend, xend);

return 0;

}

int shmarkli(Bitmap\* bmp, int ak, int xend, int yend)

{

if(f\_sop\_ver == 1) {

for(int i = 0; i < V\_LIN; i++)

sopli[i] = 1;

f\_sop = 1;

f\_sop\_ver = 0;

}

xend = yend;

Graphics graphics(bmp);

GraphicsPath kb1;

kb1.AddLine(X(0.99), Y(0.01), X(0.99), Y(0.99));

kb1.AddLine(X(0.01), Y(0.99), X(0.01), Y(0.01));

SolidBrush farb\_2(Color(255, 255, 255, 255));

Matrix myMatrix;

myMatrix.Translate((float)X(f\_sop), (float)Y(1));

for(int i = 0; i < V\_LIN; i++) {

kb1.Transform(&myMatrix);

if(sopli[i] && f\_sop != (H\_LIN + 1)) {

graphics.FillPath(&farb\_2, &kb1);

}

myMatrix.Reset();

myMatrix.Translate((float)X(0), (float)Y(1));

}

if(!(f\_sop = (f\_sop == (H\_LIN + 1) ? 0 : f\_sop + 1))) {

SetTimer(hhhwww, IDT\_TIMER4, 200000, (TIMERPROC)NULL);

if(typ\_gry == 0 && f\_next\_level == 1) {

f\_next\_level = 0;

setup\_1(typ\_gry);

}

InvalidateRect(hhhwww,NULL, false);

UpdateWindow(hhhwww);

}

return 0;

}