Пензенский государственный университет

Кафедра «Системы автоматизированного проектирования»

**ОТЧЕТ**

по лабораторной работе №1

по дисциплине: «Компьютерная графика и 3D моделирование.»

на тему: «Векторное представление графических объектов. Аффинные представления.»

Выполнил:

студент группы 21ВВC1

Нагорная Д. А.

Приняли:

д. т. н., профессор Бождай А. С.

к. т. н., доцент Финогеев А. А.

Пенза, 2023

**Цель работы:** изучение моделей, методов и средств представления, визуализации и преобразования двухмерных векторных графических объектов, знакомство с аффинными преобразованиями на плоскости.

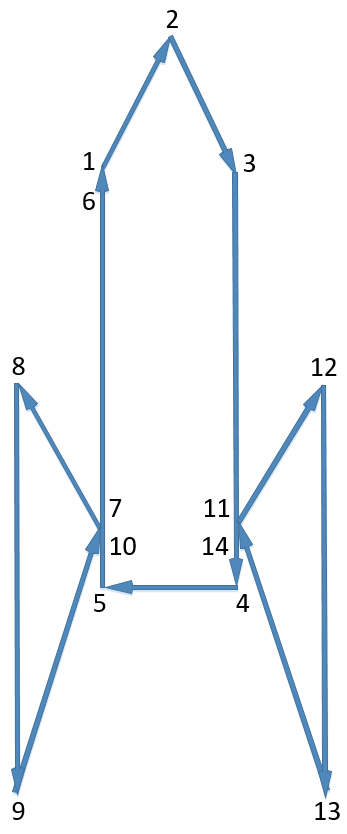
**Порядок выполнения работы:**

1. Согласно заданию, составить описание графического объекта путем перечисления векторов координат базовых точек и последовательности соединяющих их отрезков. Размеры объекта и координаты точек подбираются произвольно, с учетом наиболее оптимального положения на экране как самого объекта, так и его копий, полученных в результате аффинных преобразований.
2. Разработать программу, реализующую процесс визуализации заданного графического объекта с помощью средств GDI.
3. Составить матрицы аффинных преобразований в соответствии с заданием.
4. Добавить в программу процедуру, реализующую заданные аффинные преобразования графического объекта с помощью матричных операций.

**Вариант №5.**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Название | Графический объект | Аффинные преобразования |
| 5 | Ракета |  | - перенос x-1 дл., y+2 выс.  - уменьшение по x,y в 4 раза  - отображение отн. x=y |

**Модель графического объекта** в виде описания координат точек и соединяющих их отрезков:



//Количесвто вершин

const int NumbVertex = 14;

//Координаты фигуры

double myCoord[14][4] = { {1050, 125, 1, 0}, //1

{1070, 90, 1, 1}, //2

{1090, 125, 1, 1}, //3

{1090, 280, 1, 1}, //4

{1050, 280, 1, 1}, //5

{1050, 125, 1, 1}, //6

{1050, 260, 1, 0}, //7

{1030, 220, 1, 1}, //8

{1030, 350, 1, 1}, //9

{1050, 260, 1, 1}, //10

{1090, 260, 1, 0}, //11

{1110, 220, 1, 1}, //12

{1110, 350, 1, 1}, //13

{1090, 260, 1, 1} }; //14

**Матрицы аффинных преобразований:**

//перемещение (-1; 2)

double mTrans1[3][3] = { {1, 0, 0},

{0, 1, 0},

{-80, 160, 1} };

//масштабирование (уменьшение объекта в 4 раза)

double mTrans2[3][3] = { {0.25, 0, 0},

{0, 0.25, 0},

{0, 0, 1} };

//зеркальное отображение (относительно прямой y=x)  
double mTrans3[3][3] = { {0, 1, 0},  
 {1 ,0, 0},  
 {0, 0, 1} };

**Листинг программы:**

Файл lab1.cpp

// lab1.cpp : Определяет точку входа для приложения.

//

#include "framework.h"

#include "lab1.h"

#define MAX\_LOADSTRING 100

// Глобальные переменные:

HINSTANCE hInst; // текущий экземпляр

WCHAR szTitle[MAX\_LOADSTRING]; // Текст строки заголовка

WCHAR szWindowClass[MAX\_LOADSTRING]; // имя класса главного окна

//Количесвто вершин

const int NumbVertex = 14;

//Координаты фигуры

double myCoord[14][4] = { {1050, 125, 1, 0}, //1

{1070, 90, 1, 1}, //2

{1090, 125, 1, 1}, //3

{1090, 280, 1, 1}, //4

{1050, 280, 1, 1}, //5

{1050, 125, 1, 1}, //6

{1050, 260, 1, 0}, //7

{1030, 220, 1, 1}, //8

{1030, 350, 1, 1}, //9

{1050, 260, 1, 1}, //10

{1090, 260, 1, 0}, //11

{1110, 220, 1, 1}, //12

{1110, 350, 1, 1}, //13

{1090, 260, 1, 1} }; //14

//Аффинные преобразования

//перемещение (-1; 2)

double mTrans1[3][3] = { {1, 0, 0},

{0, 1, 0},

{-80, 160, 1} };

//масштабирование (уменьшение объекта в 4 раза)

double mTrans2[3][3] = { {0.25, 0, 0},

{0, 0.25, 0},

{0, 0, 1} };

//зеркальное отображение (относительно прямой y=x)  
double mTrans3[3][3] = { {0, 1, 0},  
 {1 ,0, 0},  
 {0, 0, 1} };

//Умножение матриц

void Multiply\_matrix(HWND hWnd, double mTrans[3][3]);

//Дополнительная матрица для пункта "отрисовать заново"

void two\_matrix(HWND hWnd);

// Отправить объявления функций, включенных в этот модуль кода:

ATOM MyRegisterClass(HINSTANCE hInstance);

BOOL InitInstance(HINSTANCE, int);

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

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

int APIENTRY wWinMain(\_In\_ HINSTANCE hInstance,

\_In\_opt\_ HINSTANCE hPrevInstance,

\_In\_ LPWSTR lpCmdLine,

\_In\_ int nCmdShow)

{

UNREFERENCED\_PARAMETER(hPrevInstance);

UNREFERENCED\_PARAMETER(lpCmdLine);

// TODO: Разместите код здесь.

// Инициализация глобальных строк

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

LoadStringW(hInstance, IDC\_LAB1, szWindowClass, MAX\_LOADSTRING);

MyRegisterClass(hInstance);

// Выполнить инициализацию приложения:

if (!InitInstance (hInstance, nCmdShow))

{

return FALSE;

}

HACCEL hAccelTable = LoadAccelerators(hInstance, MAKEINTRESOURCE(IDC\_LAB1));

MSG msg;

// Цикл основного сообщения:

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

{

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

{

TranslateMessage(&msg);

DispatchMessage(&msg);

}

}

return (int) msg.wParam;

}

//

// ФУНКЦИЯ: MyRegisterClass()

//

// ЦЕЛЬ: Регистрирует класс окна.

//

ATOM MyRegisterClass(HINSTANCE hInstance)

{

WNDCLASSEXW 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 = LoadIcon(hInstance, MAKEINTRESOURCE(IDI\_LAB1));

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

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

wcex.lpszMenuName = MAKEINTRESOURCEW(IDC\_LAB1);

wcex.lpszClassName = szWindowClass;

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

return RegisterClassExW(&wcex);

}

//

// ФУНКЦИЯ: InitInstance(HINSTANCE, int)

//

// ЦЕЛЬ: Сохраняет маркер экземпляра и создает главное окно

//

// КОММЕНТАРИИ:

//

// В этой функции маркер экземпляра сохраняется в глобальной переменной, а также

// создается и выводится главное окно программы.

//

BOOL InitInstance(HINSTANCE hInstance, int nCmdShow)

{

hInst = hInstance; // Сохранить маркер экземпляра в глобальной переменной

HWND hWnd = CreateWindowW(szWindowClass, szTitle, WS\_OVERLAPPEDWINDOW,

CW\_USEDEFAULT, 0, CW\_USEDEFAULT, 0, nullptr, nullptr, hInstance, nullptr);

if (!hWnd)

{

return FALSE;

}

ShowWindow(hWnd, SW\_SHOWMAXIMIZED); // показываем окно на весь экран

UpdateWindow(hWnd);

return TRUE;

}

//

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

//

// ЦЕЛЬ: Обрабатывает сообщения в главном окне.

//

// WM\_COMMAND - обработать меню приложения

// WM\_PAINT - Отрисовка главного окна

// WM\_DESTROY - отправить сообщение о выходе и вернуться

//

//

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

{

switch (message)

{

case WM\_COMMAND:

{

int wmId = LOWORD(wParam);

// Разобрать выбор в меню:

switch (wmId)

{

case IDM\_ABOUT:

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

break;

case IDM\_EXIT:

DestroyWindow(hWnd);

break;

case IDM\_TRANSLATION:

Multiply\_matrix(hWnd, mTrans1);

InvalidateRect(hWnd, NULL, TRUE);

UpdateWindow(hWnd);

break;

case IDM\_DILATATION:

Multiply\_matrix(hWnd, mTrans2);

InvalidateRect(hWnd, NULL, TRUE);

UpdateWindow(hWnd);

break;

case IDM\_REFLECTION:

Multiply\_matrix(hWnd, mTrans3);

InvalidateRect(hWnd, NULL, TRUE);

UpdateWindow(hWnd);

break;

case IDM\_AGAIN:

two\_matrix(hWnd);

InvalidateRect(hWnd, NULL, TRUE);

UpdateWindow(hWnd);

break;

default:

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

}

}

break;

case WM\_PAINT:

{

PAINTSTRUCT ps;

HDC hdc = BeginPaint(hWnd, &ps);

// TODO: Добавьте сюда любой код прорисовки, использующий HDC...

//

for (int i = 0; i < 14; i++)

{

if (myCoord[i][3] == 0)

{

MoveToEx(hdc, myCoord[i][0], myCoord[i][1], NULL);

}

else if (myCoord[i][3] == 1)

{

LineTo(hdc, myCoord[i][0], myCoord[i][1]);

}

}

EndPaint(hWnd, &ps);

}

break;

case WM\_DESTROY:

PostQuitMessage(0);

break;

default:

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

}

return 0;

}

// Обработчик сообщений для окна "О программе".

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;

}

void Multiply\_matrix(HWND hWnd, double mTrans[3][3])

{

double newmatrix[NumbVertex][4] = { {0, 0, 0, 0},

{0, 0, 0, 1},

{0, 0, 0, 1},

{0, 0, 0, 1},

{0, 0, 0, 1},

{0, 0, 0, 1},

{0, 0, 0, 0},

{0, 0, 0, 1},

{0, 0, 0, 1},

{0, 0, 0, 1},

{0, 0, 0, 0},

{0, 0, 0, 1},

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

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

{

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

{

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

{

newmatrix[i][j] += myCoord[i][k] \* mTrans[k][j];

}

}

}

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

{

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

{

myCoord[i][j] = newmatrix[i][j];

}

}

}

void two\_matrix(HWND hWnd)

{

double newmatrix[14][4] = { {1050, 125, 1, 0}, //1

{1070, 90, 1, 1}, //2

{1090, 125, 1, 1}, //3

{1090, 280, 1, 1}, //4

{1050, 280, 1, 1}, //5

{1050, 125, 1, 1}, //6

{1050, 260, 1, 0}, //7

{1030, 220, 1, 1}, //8

{1030, 350, 1, 1}, //9

{1050, 260, 1, 1}, //10

{1090, 260, 1, 0}, //11

{1110, 220, 1, 1}, //12

{1110, 350, 1, 1}, //13

{1090, 260, 1, 1} }; //14

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

{

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

{

myCoord[i][j] = newmatrix[i][j];

}

}

}

Файл Resource.h

#define IDS\_APP\_TITLE 103

#define IDR\_MAINFRAME 128

#define IDD\_LAB1\_DIALOG 102

#define IDD\_ABOUTBOX 103

#define IDM\_ABOUT 104

#define IDM\_EXIT 105

#define IDM\_TRANSLATION 50

#define IDM\_DILATATION 51

#define IDM\_REFLECTION 52

#define IDM\_AGAIN 53

#define IDI\_LAB1 107

#define IDI\_SMALL 108

#define IDC\_LAB1 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

Файл lab1.rc

#include "resource.h"

#define APSTUDIO\_READONLY\_SYMBOLS

#ifndef APSTUDIO\_INVOKED

#include "targetver.h"

#endif

#define APSTUDIO\_HIDDEN\_SYMBOLS

#include "windows.h"

#undef APSTUDIO\_HIDDEN\_SYMBOLS

#undef APSTUDIO\_READONLY\_SYMBOLS

#if !defined(AFX\_RESOURCE\_DLL) || defined(AFX\_TARG\_RUS)

LANGUAGE 25, 1

IDI\_LAB1 ICON "lab1.ico"

IDI\_SMALL ICON "small.ico"

IDC\_LAB1 MENU

BEGIN

POPUP "&Файл"

BEGIN

MENUITEM "Вы&ход", IDM\_EXIT

END

POPUP "&Действия"

BEGIN

MENUITEM "&Перенос x-1 дл., y+2 выс.", IDM\_TRANSLATION

MENUITEM "&Уменьшение по x, y в 4 раза", IDM\_DILATATION

MENUITEM "&Отображение отн. x=y", IDM\_REFLECTION

MENUITEM "&Отрисовать заново", IDM\_AGAIN

END

POPUP "&Справка"

BEGIN

MENUITEM "&О программе...", IDM\_ABOUT

END

END

IDC\_LAB1 ACCELERATORS

BEGIN

"?", IDM\_ABOUT, ASCII, ALT

"/", IDM\_ABOUT, ASCII, ALT

END

IDD\_ABOUTBOX DIALOGEX 0, 0, 170, 62

STYLE DS\_SETFONT | DS\_MODALFRAME | DS\_FIXEDSYS | WS\_POPUP | WS\_CAPTION | WS\_SYSMENU

CAPTION "О программе lab1"

FONT 8, "MS Shell Dlg"

BEGIN

ICON IDR\_MAINFRAME,IDC\_STATIC,14,14,21,20

LTEXT "lab1, Версия 1.0",IDC\_STATIC,42,14,114,8,SS\_NOPREFIX

LTEXT "(c) 2023",IDC\_STATIC,42,26,114,8

DEFPUSHBUTTON "ОК",IDOK,113,41,50,14,WS\_GROUP

END

#ifdef APSTUDIO\_INVOKED

GUIDELINES DESIGNINFO

BEGIN

IDD\_ABOUTBOX, DIALOG

BEGIN

LEFTMARGIN, 7

RIGHTMARGIN, 163

TOPMARGIN, 7

BOTTOMMARGIN, 55

END

END

#endif // APSTUDIO\_INVOKED

#ifdef APSTUDIO\_INVOKED

1 TEXTINCLUDE

BEGIN

"resource.h\0"

END

2 TEXTINCLUDE

BEGIN

"#ifndef APSTUDIO\_INVOKED\r\n"

"#include ""targetver.h""\r\n"

"#endif\r\n"

"#define APSTUDIO\_HIDDEN\_SYMBOLS\r\n"

"#include ""windows.h""\r\n"

"#undef APSTUDIO\_HIDDEN\_SYMBOLS\r\n"

"\0"

END

3 TEXTINCLUDE

BEGIN

"\r\n"

"\0"

END

#endif // APSTUDIO\_INVOKED

STRINGTABLE

BEGIN

IDC\_LAB1 "LAB1"

IDS\_APP\_TITLE "lab1"

END

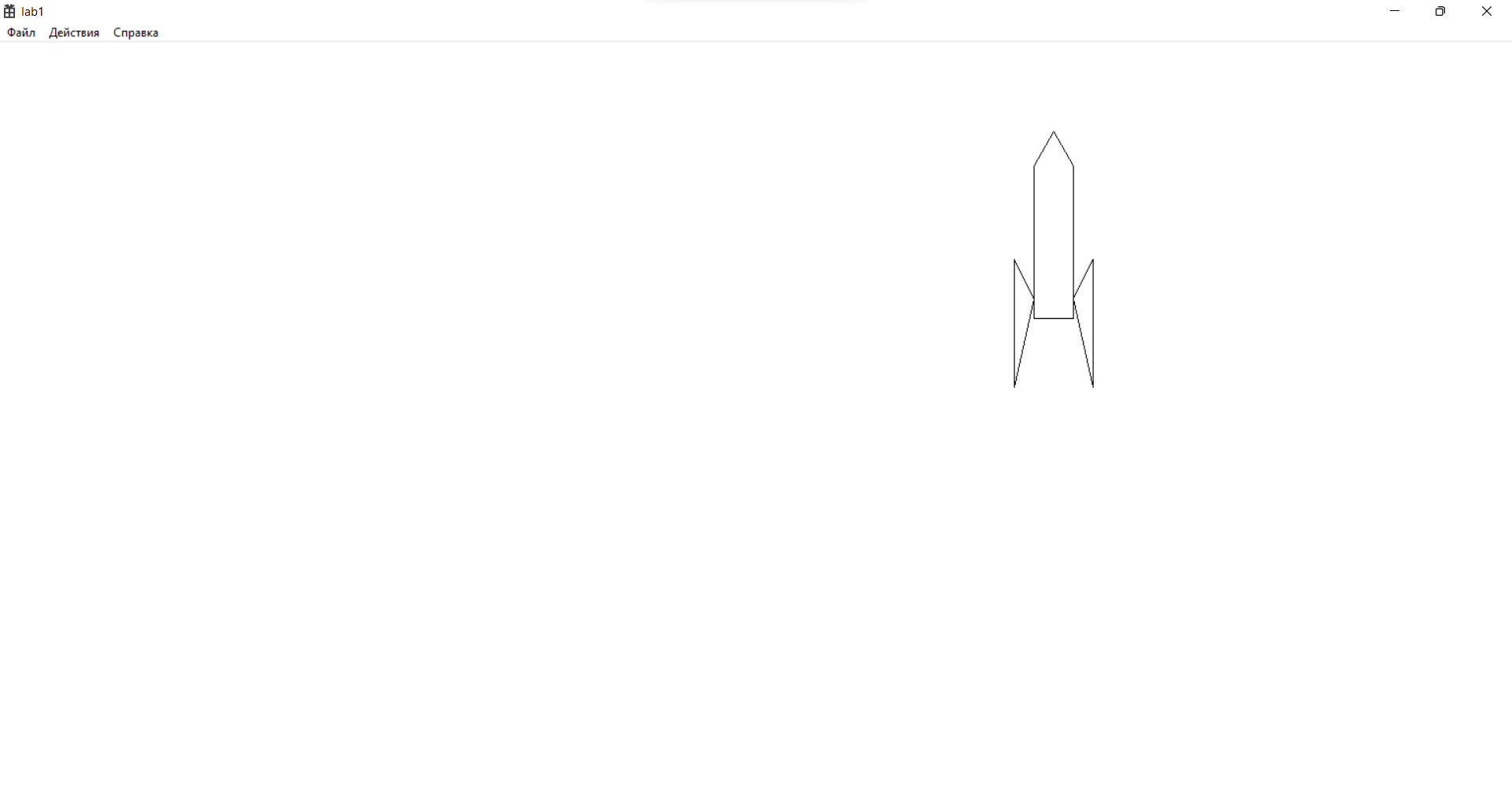
#endif

#ifndef APSTUDIO\_INVOKED

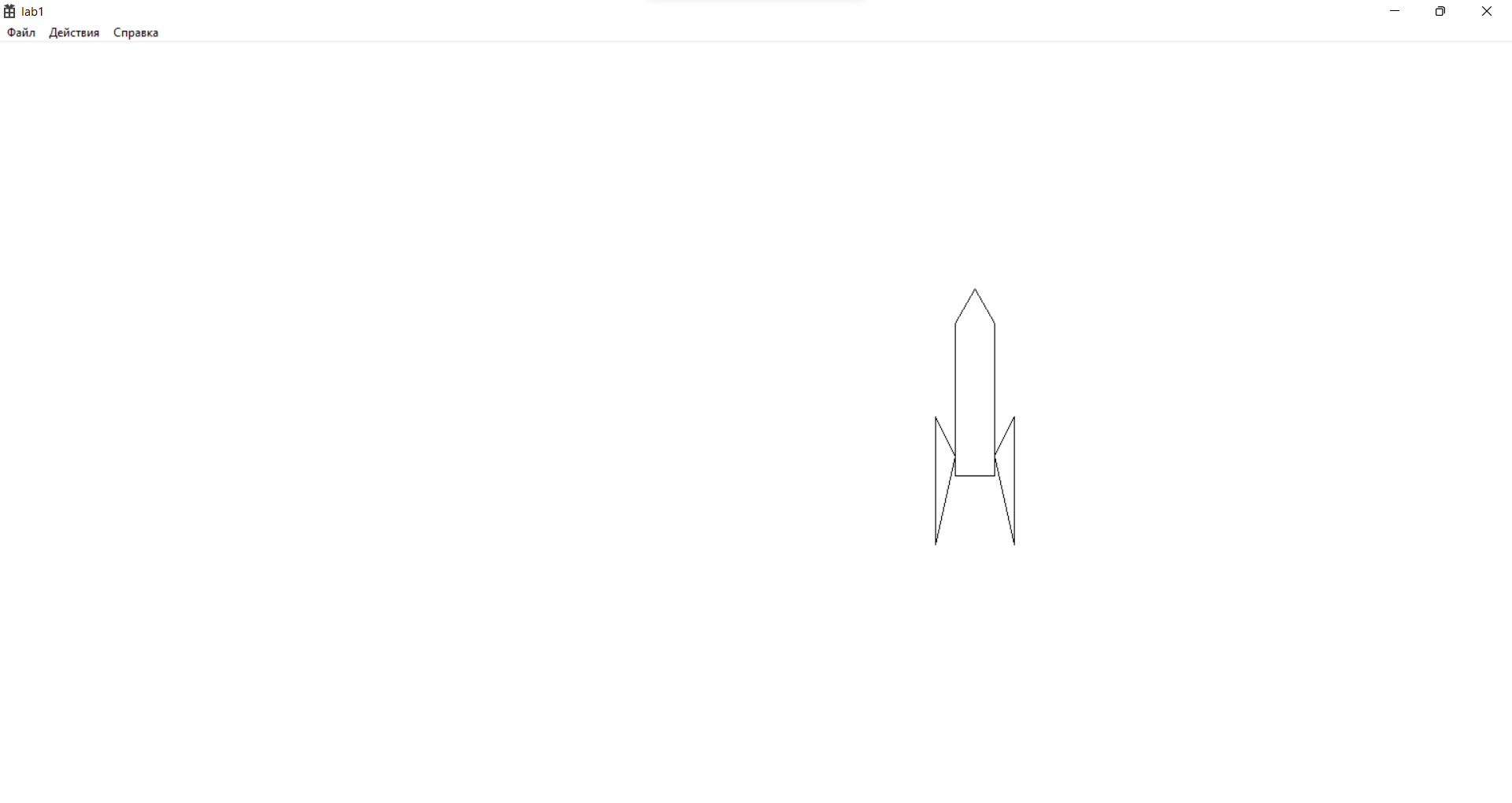
#endif // not APSTUDIO\_INVOKED

**Машинная распечатка экрана с изображением графического объекта и всех его выполненных аффинных преобразований:**

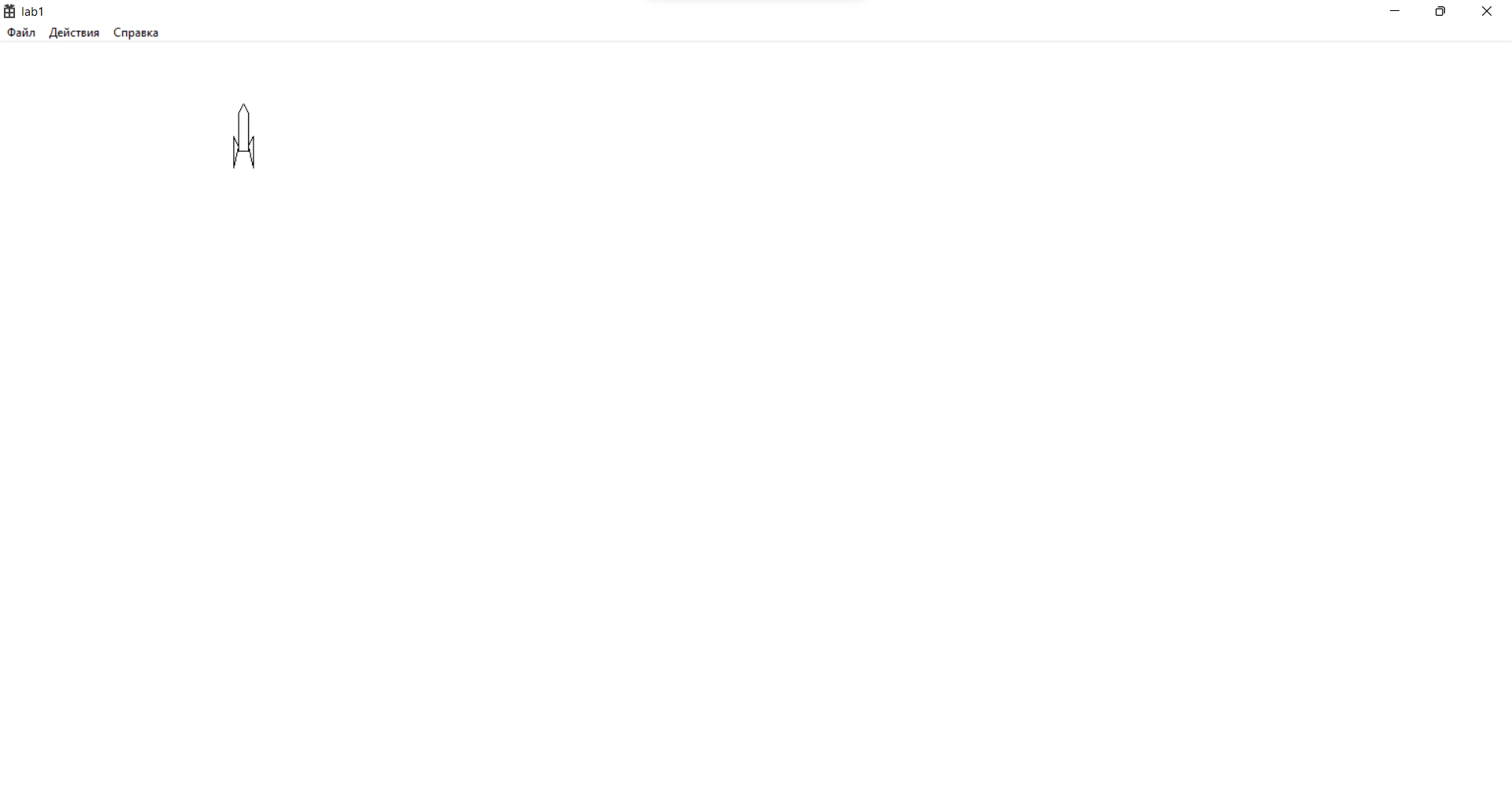
Приложение А. «».



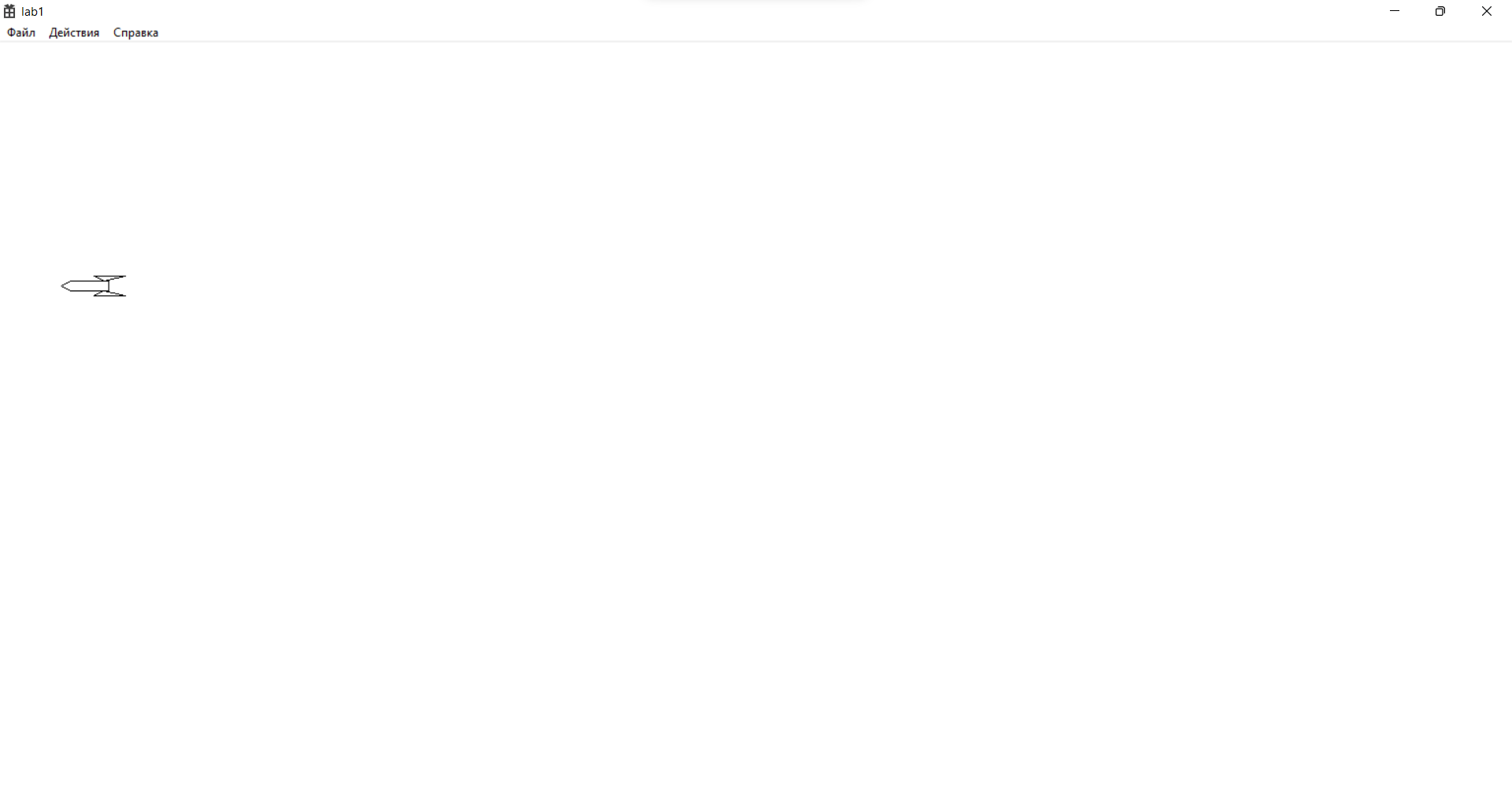
Приложение Б. «Перемещение».



Приложение В. «Масштабирование».



Приложение Г. «Зеркальное отображение».



**Вывод:** в ходе данной лабораторной работы были изучены модели, методы и средства представления, визуализации и преобразования двухмерных векторных графических объектов, ознакомились с аффинными преобразованиями на плоскости; были изучены функции для работы с графическим интерфейсом Windows GPI (Graphic Device Interface), функции по работе с векторной графикой (GetDC, MoveToEx, LineTo и т.д.); изучено представление экранной системы координат, используемой для вывода изображений, представление точки в виде вектора с темя значениями.

//система координат:

MoveToEx(hdc, width + 30, height + 40, NULL);

LineTo(hdc, width + 30, height \* 2 + 40);

LineTo(hdc, width \* 2 + 30, height \* 2 + 40);

int aver\_sum = 0, aver\_per\_pix = 0, aver\_color = 0;

for (int i = width + 30; i < width \* 2 + 30; i++)

{

aver\_sum = 0;

MoveToEx(hdc, i, 320, NULL);

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

{

color = GetPixel(hdc, i, j);

aver\_per\_pix = (GetRValue(color) + GetGValue(color) + GetBValue(color)) / 3;

aver\_sum += aver\_per\_pix; //сумма пикселей каждого столбца

//высота столбика равна aver\_sum, а по x их размещать на каждом пикселе

}

aver\_color = aver\_sum / height;//среднее арифм. каждого столбца

LineTo(hdc, i, (aver\_sum / 100) + height);

}

#include <windows.h>

HBITMAP bmp;

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

HWND button1, button2, button3, button4, button5, orig;

HDC Hdc;

int rezcolors[3][3];

int newcolors;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,

                    PSTR szCmdLine, int iCmdShow)

     {

     static char szAppName[] = "Win" ;

     HWND        hwnd ;

     MSG         msg ;

     WNDCLASSEX  wndclass ;

     wndclass.cbSize        = sizeof (wndclass) ;

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

     wndclass.lpfnWndProc   = WndProc ;

     wndclass.cbClsExtra    = 0 ;

     wndclass.cbWndExtra    = 0 ;

     wndclass.hInstance     = hInstance ;

     wndclass.hIcon         = LoadIcon (NULL, IDI\_APPLICATION) ;

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

     wndclass.hbrBackground = (HBRUSH) GetStockObject (WHITE\_BRUSH) ;

     wndclass.lpszMenuName  = NULL ;

     wndclass.lpszClassName = szAppName ;

     wndclass.hIconSm       = LoadIcon (NULL, IDI\_APPLICATION) ;

     RegisterClassEx (&wndclass) ;

     hwnd = CreateWindow (szAppName,         // window class name

                    "Smoothing",                 // window caption

                    WS\_OVERLAPPEDWINDOW,     // window style

                    CW\_USEDEFAULT,           // initial x position

                    CW\_USEDEFAULT,           // initial y position

                    600,                    // initial x size

                    600,                    // initial y size

                    NULL,                    // parent window handle

                    NULL,                    // window menu handle

                    hInstance,               // program instance handle

                    NULL) ;                  // creation parameters

     ShowWindow (hwnd, iCmdShow) ;

     UpdateWindow (hwnd) ;

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

          {

          TranslateMessage (&msg) ;

          DispatchMessage (&msg) ;

          }

     return msg.wParam ;

     }

void gist(int xx)

{

    Rectangle(Hdc,xx-1,350,xx+186,350);

    int r,g,b,spectr,sumspectr,k2;

    for (int x=xx;x<xx+250;x++)

    {

        sumspectr=0;

        for (int y=100;y<350;y++)

        {

            r=GetRValue(GetPixel(Hdc, x, y));

        //  g=GetGValue(GetPixel(Hdc, x, y));

            //b=GetBValue(GetPixel(Hdc, x, y));

            spectr=r;

            sumspectr+=spectr;

        }

        k2=1;

        sumspectr=sumspectr/500;

        for (int q=0;q<sumspectr;q++)

        {

            SetPixel(Hdc, x, 350-k2, RGB(0,0,0));

            k2++;

        }

    }

}

LRESULT CALLBACK WndProc (HWND hwnd, UINT iMsg, WPARAM wParam, LPARAM lParam)

     {

     HDC         hdc=GetDC(hwnd);

     PAINTSTRUCT ps ;

     RECT        rect ;

     Hdc=hdc;

     TCHAR buf1[256];

     TCHAR buf2[256];

     char buf111[256];

     int buf11;

     int buf22;

    CharToOem(buf1,buf111);

     buf11=int(buf111);

     buf22=int(buf2);

     switch (iMsg)

          {

          case WM\_CREATE :

              bmp=(HBITMAP) LoadImage(NULL, "f.bmp", IMAGE\_BITMAP,0,0, LR\_LOADFROMFILE);

              button1 = CreateWindow ("EDIT","255",SS\_CENTER | WS\_CHILD | WS\_VISIBLE,150,50,150,50,hwnd,NULL,0,NULL);

              button2 = CreateWindow ("BUTTON","перерисовка",WS\_CHILD | WS\_VISIBLE,300,50,150,50,hwnd,NULL,0,NULL);

              button3 = CreateWindow ("BUTTON","Ввести нижнюю границу",WS\_CHILD | WS\_VISIBLE,0,0,150,50,hwnd,NULL,0,NULL);

              button4 = CreateWindow ("BUTTON","Ввести верхнюю границу",WS\_CHILD | WS\_VISIBLE,150,0,150,50,hwnd,NULL,0,NULL);

              button5 = CreateWindow ("BUTTON","гистограмма",WS\_CHILD | WS\_VISIBLE,300,0,150,50,hwnd,NULL,0,NULL);

              orig = CreateWindow("EDIT","0", SS\_CENTER | WS\_CHILD | WS\_VISIBLE, 0, 50, 150, 50, hwnd, NULL, 0, NULL);

              return 0 ;

          case WM\_COMMAND:

*/\*    if (lParam==LPARAM(button1))*

*{*

*TCHAR buf1[256];*

*GetWindowText(orig,buf1,lstrlen(buf1));*

*TCHAR b1[256];*

*GetWindowText(button5,b1,lstrlen(b1));*

*}\*/*

              if (lParam==LPARAM(button2))

              {

                  for (int x=1;x<250;x++)

                      for (int y=101;y<287;y++)

                  {

                      newcolors=GetRValue(GetPixel(hdc, x-1, y-1));

                      if(buf11>=newcolors)

                      {

                          SetPixel(hdc, x, y, RGB(0,0,0));

                      };

                      if(buf22<=newcolors)

                      {

                      SetPixel(hdc, x, y, RGB(255,255,255));

                      };

                      if(buf11<=newcolors && buf22>=newcolors)

                      {

                      newcolors=newcolors-buf11;

                      SetPixel(hdc,x,y,RGB(newcolors,newcolors,newcolors));

                      }

                      }

              }

              if (lParam==LPARAM(button3))

              {

                  GetWindowText(orig,buf1,lstrlen(buf1));

              }

              if (lParam==LPARAM(button4))

              {

                   GetWindowText(button1,buf2,lstrlen(buf2));

              }

              if (lParam==LPARAM(button5))

              {

                  gist(0);

              }

              return 0;

          case WM\_PAINT :

               hdc = BeginPaint (hwnd, &ps) ;

               HDC memdc;

               memdc=CreateCompatibleDC(hdc);

               SelectObject(memdc,bmp);

               BitBlt(hdc, 0, 100, 250, 186, memdc, 0, 0, SRCCOPY);

               DeleteDC(memdc);

               EndPaint (hwnd, &ps) ;

               return 0 ;

          case WM\_DESTROY :

               PostQuitMessage (0) ;

               return 0 ;

          }

     return DefWindowProc (hwnd, iMsg, wParam, lParam) ;

     }