Министерство образования и науки Российской Федерации

Курский государственный университет

Кафедра ПОиАИС

**Аффинные преобразования на плоскости**

отчет о выполнении лабораторной работы №2

по дисциплине «Компьютерная графика»

Выполнил студент гр. 313 Л.А. Сергеев

Проверил доц. Е.А. Сухотерин

Курск – 2020

1 Цель работы

Изучение математических методов аффинных преобразований на плоскости и практическое освоение приемов программной реализации аффинных преобразований.

2 Задание

На самостоятельно выбранном языке программирования создать программу, реализующую преобразования фигуры (многоугольника) на плоскости в соответствии с индивидуальным вариантом №6.

Преобразования:

1. вдоль заданной стороны фигуры (указание стороны, расстояние переноса равно длине стороны);
2. вдоль оси OX (ввод масштабного коэффициента);
3. относительно прямой с направляющим вектором OP, проходящей через начало координат (указание точки P);
4. вокруг заданной точки плоскости (указание точки, ввод величины угла поворота).

Под термином «ввод» понимается получение программой информации от пользователя в числовом виде, под термином «указание» – получение программой информации от пользователя посредством выбора соответствующего элемента на рисунке при помощи указателя мыши.

Программа должна:

1. включать два окна (на одной форме или на разных), содержащих изображение фигуры до и после преобразования и управляющие элементы для ввода необходимой информации;
2. отображать координаты вершин исходной и преобразованной фигуры;
3. отображать итоговую матрицу преобразования и обеспечивать возможность внесения ручных изменений в эту матрицу и применения полученной матрицы к исходной фигуре;
4. иметь возможность отображать или скрывать координатные оси и масштабную сетку в окнах вывода фигур;
5. обеспечивать ввод фигуры для преобразования при помощи мыши посредством указания вершин фигуры;
6. для каждого преобразования обеспечивать возможность динамической визуализации многократного последовательного применения заданного преобразования к исходному многоугольнику (до остановки пользователем).

3 Краткие теоретические сведения, описание и блок-схемы основных алгоритмов

Преобразование плоскости называется аффинным, если существуют такие две аффинные системы координат, что координаты любой точки в первой системе совпадают с координатами ее образа во второй системе.

Любое линейное преобразование может быть представлено в матричном виде:

P′ = P \* A + B,

где P = (x, y) – координаты исходной точки (прообраза), P′ = (x′, y′) – координаты точки после преобразования (образа), A ∈ R2×2, B ∈ R1×2 – матрицы преобразования.

Аффинное преобразование может быть активным, т.е. применяться к объекту или пассивным и применяться к системе координат.

Перенос, масштабирование и поворот вокруг начала координат являются элементарными аффинными преобразованиями, с помощью их комбинаций можно представить любое произвольное аффинное преобразование.

Для снижения вычислительных затрат при реализации аффинных преобразований удобнее ввести однородные координаты.

Однородные координаты точки плоскости P с декартовыми координатами (x, y) – это тройка вида C = (a, b, c) = (x \* h, y \* h, h), где h – произвольное число, отличное от нуля. В однородных координатах аффинные преобразования имеют одинаковую форму произведения вектора исходных координат на матрицу преобразования:

C′ = C \* A.

Блок-схема реализации алгоритма переноса вдоль стороны (Aв;Bв) представлена на рисунке 1.



Рисунок 1 - Алгоритм переноса вдоль стороны Xв Yв.

Блок-схема реализации алгоритма отражение относительно прямой с направляющим вектором OP, проходящей через начало координат представлена на рисунке 2.



Рисунок 2 - Алгоритм отражения относительно прямой с направляющим вектором OP, проходящей через начало координат



Рисунок 2 - Продолжение

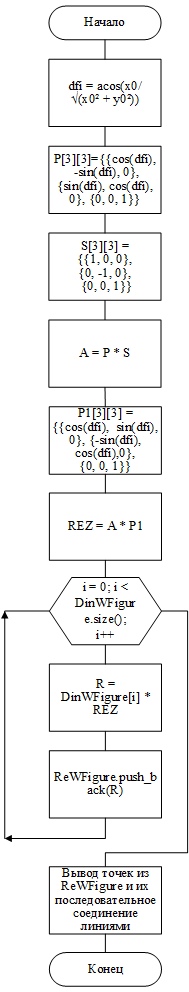


Рисунок 3 – Алгоритм отражения относительно Ox.

4 Листинг программы

#include <vcl.h>

#include <math.h>

#include <vector>

#include <string>

using namespace std;

#pragma hdrstop

#include <cmath>

#include "Unit1.h"

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

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TForm1 \*Form1;

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

\_\_fastcall TForm1::TForm1(TComponent\* Owner)

: TForm(Owner)

{

}

TPoint Points[25];

bool proverka=false;//проверка смежности вершин

bool printpr=false;//проверка флага рисования

bool ter=false;// для переноса

bool flpr=false;//для прямой)

double dfi;//угол

bool bdf=false;//для угла

int prosm=0;//для переноса

int nX, nY; //размер Канвы

int Xnull, Ynull; //начало координат

double Xmin, Xmax, Ymin, Ymax; //границы

int dX, dY; //масштаб

double AlfX, AlfY; //альфа

double Matrix[3][3];

vector <TPoint> Figure; //кординаты фигуры

vector <TPoint> ReFigure; //кординаты изменённой фигуры

//vector <TPoint> DinFigure; //динамическа фигура

struct WPoint //строктура точки в мировых

{

double x;

double y;

WPoint (double X, double Y)

{

x = X;

y = Y;

}

};

vector <WPoint> WFigure; //кординаты фигуры в мировых

vector <WPoint> ReWFigure; //кординаты изменённой фигуры в мировых

vector <WPoint> DinWFigure; //динамическая фигура в мировых

TPoint Tck; //выбранная точка

TPoint Tck2; //выбранная точка 2 для переноса по стороне

TPoint Prymaya;//точка для прямой

double OXrate; //коэф. масстабирования

int TopF=0,TopF2=0; //выбранная вершина

double Turn; //угол повторота

bool FlagPaint = false; //флаг рисования по матрице

bool StartPaint = false; //флаг рисования в динамике

bool PaintEnd = false; //флаг остановки рисования в динамике

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

int W2Sx (double Xm) //К экранным Х

{

int Xe = round(AlfX\*Xm)+Xnull;

return Xe;

}

int W2Sy (double Ym) //К экранным У

{

int Ye = (-1) \* round(AlfY\*Ym)+Ynull;

return Ye;

}

double S2Wx (int Xe) //К мировым Х

{

double Xm = (Xe-Xnull)/AlfX;

return Xm;

}

double S2Wy (int Ye) //К мировым Y

{

double Ym = -(Ye-Ynull)/AlfY;

return Ym;

}

//рисование линии Мировые

void LineW (TImage \*I, double x1, double y1, double x2, double y2)

{

I->Canvas->MoveTo(W2Sx(x1),W2Sy(y1));

I->Canvas->LineTo(W2Sx(x2),W2Sy(y2));

}

//рисование линии Экранные

void LineS (TImage \*I, double x1, double y1, double x2, double y2)

{

I->Canvas->MoveTo(x1,y1);

I->Canvas->LineTo(x2,y2);

}

//рисование осей

void PaintAxis (TImage \*I)

{

I->Canvas->Pen->Width = 2;

//ось Х

LineS(I,0,Ynull,nX,Ynull);

LineS(I,nX-15,Ynull+5,nX,Ynull);

LineS(I,nX-15,Ynull-5,nX,Ynull);

I->Canvas->TextOut(nX-10,Ynull-7-I->Canvas->TextHeight("X"),"X");

//ось У

LineS(I,Xnull,0,Xnull,nY);

LineS(I,Xnull-5,0+15,Xnull,0);

LineS(I,Xnull+5,0+15,Xnull,0);

I->Canvas->TextOut(Xnull-12,0+15-I->Canvas->TextHeight("Y"),"Y");

//подписть О

I->Canvas->TextOut(Xnull-12,Ynull+15-I->Canvas->TextHeight(")"),"O");

I->Canvas->Pen->Width = 1;

}

//рисование сетки

void PaintGrid (TImage \*I)

{

I->Canvas->Pen->Style = psDot;

//линии Х

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

LineW (I, Xmin + (i \* dX), Ymin, Xmin + (i \* dX), Ymax);

//линии У

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

LineW (I, Xmin, Ymin + (i \* dY), Xmax, Ymin + (i \* dY));

I->Canvas->Pen->Style = psSolid;

}

//рисование линейки

void PaintScale (TImage \*I)

{

I->Canvas->Pen->Width = 2;

//длина линий меток

float Ys = Ymax / 30;

float Xs = Xmax / 50;

//линейка Х

for (int i = 1;i<-Xmin+Xmax;i++)

{

if (Xmin + (i \* dX) == 0) continue; //не писать 0

LineW (I, Xmin + (i \* dX), 0+Ys, Xmin + (i \* dX), 0-Ys);

I->Canvas->TextOut (W2Sx(Xmin + (i \* dX)), W2Sy(0)+7, FloatToStrF(Xmin + (i \* dX),ffFixed,10,0));

}

//линейка У

for (int i = 0;i<-Ymin+Ymax;i++)

{

if (Ymin + (i \* dY) == 0) continue; //не писать 0

LineW (I, 0+Xs, Ymin + (i \* dY), 0-Xs, Ymin + (i \* dY));

I->Canvas->TextOut (W2Sx(0)+7, W2Sy(Ymin + (i \* dY)), FloatToStrF(Ymin + (i \* dY),ffFixed,10,0));

}

I->Canvas->Pen->Width = 1;

}

//рисование фигуры мышью

void PaintFigure (TImage \*I, vector <TPoint>&Vec)

{

I->Canvas->Pen->Width = 3;

I->Canvas->Pen->Color = clRed;

I->Canvas->Brush->Style = bsClear;

I->Canvas->Font->Color=clRed;

if (Figure.size() == 1) //рисование точки

{

I->Canvas->Ellipse(Vec[0].x-1, Vec[0].y-1, Vec[0].x+1, Vec[0].y+1);

I->Canvas->TextOut (Vec[0].x+7, Vec[0].y-15, "(" +

FloatToStrF(S2Wx(Vec[0].x),ffFixed,10,2) + "; "+

FloatToStrF(S2Wy(Vec[0].y),ffFixed,10,2) + ")");

}

else //фигуры (минимум 2 точки)

{

TPoint Points[Vec.size()];

for (int i = 0; i < Vec.size(); i++)

{

Points[i] = Vec[i];

I->Canvas->TextOut (Vec[i].x+7, Vec[i].y-15, "(" +

FloatToStrF(S2Wx(Vec[i].x),ffFixed,10,2) + "; "+

FloatToStrF(S2Wy(Vec[i].y),ffFixed,10,2) + ")");

}

I->Canvas->Polygon(Points,Vec.size()-1);

}

I->Canvas->Pen->Width = 1;

I->Canvas->Pen->Color = clBlack;

I->Canvas->Brush->Style = bsSolid;

I->Canvas->Font->Color=clBlack;

}

//перенос

WPoint Transfer (WPoint P, double X, double Y) // матрица переноса

{

Matrix[0][0] = 1.00; Matrix[0][1] = 0.00; Matrix[0][2] = 0.00;

Matrix[1][0] = 0.00; Matrix[1][1] = 1.00; Matrix[1][2] = 0.00;

Matrix[2][0] = X; Matrix[2][1] = Y; Matrix[2][2] = 1.00;

double PM[3] = {P.x, P.y, 1.00};

double RM[3];

RM[0] = PM[0] \* Matrix[0][0] + PM[1] \* Matrix[1][0] + PM[2] \* Matrix[2][0];

RM[1] = PM[0] \* Matrix[0][1] + PM[1] \* Matrix[1][1] + PM[2] \* Matrix[2][1];

RM[2] = PM[0] \* Matrix[0][2] + PM[1] \* Matrix[1][2] + PM[2] \* Matrix[2][2];

P.x = RM[0];

P.y = RM[1];

return P;

}

//масштабированиеX

WPoint Scaling (WPoint P, double Xrate)

{

Matrix[0][0] = Xrate; Matrix[0][1] = 0; Matrix[0][2] = 0;

Matrix[1][0] = 0; Matrix[1][1] = 1; Matrix[1][2] = 0;

Matrix[2][0] = 0; Matrix[2][1] = 0; Matrix[2][2] = 1;

double PM[3] = {P.x, P.y, 1};

//PM[0] = StrToFloat(FloatToStrF(PM[0],ffFixed,10,5));

double RM[3];

RM[0] = PM[0] \* Matrix[0][0] + PM[1] \* Matrix[1][0] + PM[2] \* Matrix[2][0];

RM[1] = PM[0] \* Matrix[0][1] + PM[1] \* Matrix[1][1] + PM[2] \* Matrix[2][1];

RM[2] = PM[0] \* Matrix[0][2] + PM[1] \* Matrix[1][2] + PM[2] \* Matrix[2][2];

P.x = RM[0];

return P;

}

WPoint ScalingY (WPoint P, double Yrate)

{

Matrix[0][0] = 1; Matrix[0][1] = 0; Matrix[0][2] = 0;

Matrix[1][0] = 0; Matrix[1][1] = Yrate; Matrix[1][2] = 0;

Matrix[2][0] = 0; Matrix[2][1] = 0; Matrix[2][2] = 1;

double PM[3] = {P.x, P.y, 1};

double RM[3];

RM[0] = PM[0] \* Matrix[0][0] + PM[1] \* Matrix[1][0] + PM[2] \* Matrix[2][0];

RM[1] = PM[0] \* Matrix[0][1] + PM[1] \* Matrix[1][1] + PM[2] \* Matrix[2][1];

RM[2] = PM[0] \* Matrix[0][2] + PM[1] \* Matrix[1][2] + PM[2] \* Matrix[2][2];

P.x = RM[0];

P.y = RM[1];

return P;

}

WPoint TurningDf (WPoint P,double dfi1)

{

if ((dfi1>=-(M\_PI/2))&&(dfi1<=(M\_PI/2)))

{

Matrix[0][0] = cos(dfi1); Matrix[0][1] = -sin(dfi1); Matrix[0][2] = 0.00;

Matrix[1][0] = sin(dfi1); Matrix[1][1] = cos(dfi1); Matrix[1][2] = 0.00;

Matrix[2][0] = 0.00; Matrix[2][1] = 0.00; Matrix[2][2] = 1.00;

double PM[3] = {P.x, P.y, 1.00};

double RM[3];

RM[0] = PM[0] \* Matrix[0][0] + PM[1] \* Matrix[1][0] + PM[2] \* Matrix[2][0];

RM[1] = PM[0] \* Matrix[0][1] + PM[1] \* Matrix[1][1] + PM[2] \* Matrix[2][1];

RM[2] = PM[0] \* Matrix[0][2] + PM[1] \* Matrix[1][2] + PM[2] \* Matrix[2][2];

P.x = RM[0];

P.y = RM[1];

}

else

{

Matrix[0][0] = cos(2\*M\_PI-dfi1); Matrix[0][1] = -sin(2\*M\_PI-dfi1); Matrix[0][2] = 0.00;

Matrix[1][0] = sin(2\*M\_PI-dfi1); Matrix[1][1] = cos(2\*M\_PI-dfi1); Matrix[1][2] = 0.00;

Matrix[2][0] = 0.00; Matrix[2][1] = 0.00; Matrix[2][2] = 1.00;

double PM[3] = {P.x, P.y, 1.00};

double RM[3];

RM[0] = PM[0] \* Matrix[0][0] + PM[1] \* Matrix[1][0] + PM[2] \* Matrix[2][0];

RM[1] = PM[0] \* Matrix[0][1] + PM[1] \* Matrix[1][1] + PM[2] \* Matrix[2][1];

RM[2] = PM[0] \* Matrix[0][2] + PM[1] \* Matrix[1][2] + PM[2] \* Matrix[2][2];

P.x = RM[0];

P.y = RM[1];

}

return P;

}

//поворот

WPoint Turning (WPoint P, double Turn)

{

Matrix[0][0] = cos(Turn\*(M\_PI/180)); Matrix[0][1] = sin(Turn\*(M\_PI/180)); Matrix[0][2] = 0.00;

Matrix[1][0] = -sin(Turn\*(M\_PI/180)); Matrix[1][1] = cos(Turn\*(M\_PI/180)); Matrix[1][2] = 0.00;

Matrix[2][0] = 0.00; Matrix[2][1] = 0.00; Matrix[2][2] = 1.00;

double PM[3] = {P.x, P.y, 1.00};

double RM[3];

RM[0] = PM[0] \* Matrix[0][0] + PM[1] \* Matrix[1][0] + PM[2] \* Matrix[2][0];

RM[1] = PM[0] \* Matrix[0][1] + PM[1] \* Matrix[1][1] + PM[2] \* Matrix[2][1];

RM[2] = PM[0] \* Matrix[0][2] + PM[1] \* Matrix[1][2] + PM[2] \* Matrix[2][2];

P.x = RM[0];

P.y = RM[1];

return P;

}

//перевод вектора Экранных в Мировые и наоборот

void S2Wf (vector <TPoint> &S, vector <WPoint> &W)

{

W.clear();

for (int i = 0; i<S.size(); i++)

{

double X = S2Wx(S[i].x);

double Y = S2Wy(S[i].y);

WPoint WP(X,Y);

W.push\_back(WP);

}

}

void W2Sf (vector <TPoint> &S, vector <WPoint> &W)

{

S.clear();

for (int i = 0; i<W.size(); i++)

{

int X = W2Sx(W[i].x);

int Y = W2Sy(W[i].y);

TPoint WP(X,Y);

S.push\_back(WP);

}

}

//активация формы

void \_\_fastcall TForm1::FormActivate(TObject \*Sender)

{

//начальные параметры

nX = Image->Width; nY = Image->Height;

Xnull = nX / 2; Ynull = nY / 2;

Xmin = -6-1; Xmax = 6+1; Ymin = -6-1; Ymax = 6+1;

AlfX = nX /(Xmax-Xmin); AlfY = nY /(Ymax-Ymin);

dX = (Xmax-Xmin)/nX+1; dY = (Ymax-Ymin)/nY+1;

Tck = TPoint(Xnull,Ynull);

OXrate = 1;

TopF = 0;

Turn = 45;

Matrix[0][0] = 1.00; Matrix[0][1] = 0.00; Matrix[0][2] = 0.00;

Matrix[1][0] = 0.00; Matrix[1][1] = 1.00; Matrix[1][2] = 0.00;

Matrix[2][0] = 0.00; Matrix[2][1] = 0.00; Matrix[2][2] = 1.00;

BInputMClick(Sender);

Panel1->Repaint();

}

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

//перерисовка формы

void \_\_fastcall TForm1::FormPaint(TObject \*Sender)

{

Image->Canvas->Pen->Color=clBlack; //Имейдж 1

Image->Canvas->Brush->Color=clWhite;

Image->Canvas->Rectangle(0,0,nX,nY);

ImageRe->Canvas->Pen->Color=clBlack; //Имейдж 2

ImageRe->Canvas->Brush->Color=clWhite;

ImageRe->Canvas->Rectangle(0,0,nX,nY);

//прямая

if (flpr==true)

{

Image->Canvas->Pen->Color=clBlue;

Image->Canvas->Pen->Width = 2;

LineW(Image,0,0,S2Wx(Prymaya.x),S2Wy(Prymaya.y));

Image->Canvas->TextOut(Prymaya.x,Prymaya.y,"P");

ImageRe->Canvas->Pen->Color=clBlue;

ImageRe->Canvas->Pen->Width = 2;

LineW(ImageRe,0,0,S2Wx(Prymaya.x),S2Wy(Prymaya.y));

ImageRe->Canvas->TextOut(Prymaya.x,Prymaya.y,"P");

Image->Canvas->Pen->Color=clBlack;

Image->Canvas->Brush->Color=clWhite;

ImageRe->Canvas->Pen->Color=clBlack;

ImageRe->Canvas->Brush->Color=clWhite;

}

//оси и сетка

if (CAxis->Checked) {PaintAxis(Image); PaintAxis(ImageRe);}

if (CGrid->Checked) {PaintGrid(Image); PaintGrid(ImageRe);}

if (CScale->Checked) {PaintScale(Image); PaintScale(ImageRe);}

//фигура и преобразования

PaintFigure (Image, Figure);

//таймер динамики

if (SpeedButton1->Down) { Timer1->Enabled = true; PaintEnd = false;}

else

{

Timer1->Enabled = false;

//DinFigure = Figure;

S2Wf(Figure, DinWFigure);

if (StartPaint) {DinWFigure = ReWFigure; PaintEnd = true;}

}

if (RBInput->Checked) //ввод фигуры

{

PaintFigure (ImageRe, Figure);

}

if (RBVecOP->Checked && !PaintEnd) //перенос

{

ReWFigure.clear();

if (printpr==false)

{

for (int i = 0; i<DinWFigure.size(); i++)

{

ReWFigure.push\_back(Transfer(DinWFigure[i], S2Wx(Tck2.x)-S2Wx(Tck.x), S2Wy(Tck2.y)-S2Wy(Tck.y)));

}

W2Sf(ReFigure, ReWFigure);

}

}

if (RBScaling->Checked && !PaintEnd) //маcштабирование

{

Button3->Visible=true;

ReWFigure.clear();

for (int i = 0; i<DinWFigure.size(); i++)

{

ReWFigure.push\_back(Scaling(DinWFigure[i], OXrate));

}

W2Sf(ReFigure, ReWFigure);

}

if (RBReflection->Checked && !PaintEnd) //отражение

{

ReWFigure.clear();

for (int i = 0; i<DinWFigure.size(); i++)

{

WPoint P = DinWFigure[i];

if (flpr==true)

{

dfi=acos(S2Wx(Prymaya.x) / sqrt(pow(S2Wx(Prymaya.x),2)+pow(S2Wy(Prymaya.y),2)) );

P=TurningDf(P, dfi);

P=ScalingY(P,-1.00);

P=TurningDf(P, -dfi);

ReWFigure.push\_back(P);

//W2Sf(ReFigure, ReWFigure);

}

W2Sf(ReFigure, ReWFigure);

bdf=false;

}

W2Sf(ReFigure, ReWFigure);

}

if (RBTurn->Checked && !PaintEnd) //поворот

{

ReWFigure.clear();

for (int i = 0; i<DinWFigure.size(); i++)

{

WPoint P = DinWFigure[i];

P = Transfer(P, -S2Wx(Tck.x), -S2Wy(Tck.y));

P = Turning(P, Turn);

P = Transfer(P, S2Wx(Tck.x), S2Wy(Tck.y));

ReWFigure.push\_back(P);

}

W2Sf(ReFigure, ReWFigure);

//матрица поворота и переноса

double MatrixTurn[3][3];

double MatrixTrans1[3][3];

double MatrixTrans2[3][3];

double Matrix2[3][3];

MatrixTurn[0][0] = cos(Turn\*(M\_PI/180)); MatrixTurn[0][1] = sin(Turn\*(M\_PI/180)); MatrixTurn[0][2] = 0;

MatrixTurn[1][0] = -sin(Turn\*(M\_PI/180)); MatrixTurn[1][1] = cos(Turn\*(M\_PI/180)); MatrixTurn[1][2] = 0;

MatrixTurn[2][0] = 0; MatrixTurn[2][1] = 0; MatrixTurn[2][2] = 1;

MatrixTrans1[0][0] = 1; MatrixTrans1[0][1] = 0; MatrixTrans1[0][2] = 0;

MatrixTrans1[1][0] = 0; MatrixTrans1[1][1] = 1; MatrixTrans1[1][2] = 0;

MatrixTrans1[2][0] = -S2Wx(Tck.x); MatrixTrans1[2][1] = -S2Wx(Tck.x); MatrixTrans1[2][2] = 1;

MatrixTrans2[0][0] = 1; MatrixTrans2[0][1] = 0; MatrixTrans2[0][2] = 0;

MatrixTrans2[1][0] = 0; MatrixTrans2[1][1] = 1; MatrixTrans2[1][2] = 0;

MatrixTrans2[2][0] = S2Wx(Tck.x); MatrixTrans2[2][1] = S2Wx(Tck.x); MatrixTrans2[2][2] = 1;

//умножаем матрицы

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

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

{

Matrix2[i][j] = 0;

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

Matrix2[i][j] += MatrixTrans1[i][k] \* MatrixTurn[k][j];

}

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

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

{

Matrix[i][j] = 0;

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

Matrix[i][j] += Matrix2[i][k] \* MatrixTrans2[k][j];

}

}

if (((RBMatrix->Checked) && (FlagPaint)) && !PaintEnd) //по матрице

{

PaintFigure (ImageRe, ReFigure);

}

if (ReFigure.size() != 0 && !RBMatrix->Checked) //вывод преобразованной фигуры

{

PaintFigure (ImageRe, ReFigure);

BInputMClick(Sender);

}

}

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

//изменение границ Х

void \_\_fastcall TForm1::EScaleXChange(TObject \*Sender)

{

if (EScaleX->Text != "")

{

Xmin = - StrToFloat(EScaleX->Text)-1;

Xmax = StrToFloat(EScaleX->Text)+1;

AlfX = nX /(Xmax-Xmin);

dX = (Xmax-Xmin)/nX+1;

Panel1->Repaint();

}

}

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

//изменение границ У

void \_\_fastcall TForm1::EScaleYChange(TObject \*Sender)

{

if (EScaleY->Text != "")

{

Ymin = - StrToFloat(EScaleY->Text)-1;

Ymax = StrToFloat(EScaleY->Text)+1;

AlfY = nY /(Ymax-Ymin);

dY = (Ymax-Ymin)/nY+1;

Panel1->Repaint();

}

}

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

//добавление точек, ввод P, выбор вершины, ввод Е

void \_\_fastcall TForm1::ImageMouseDown(TObject \*Sender, TMouseButton Button, TShiftState Shift,

int mX, int mY)

{

if (RBInput->Checked) //ввод фигуры

{

flpr=false;

TPoint P(mX,mY);

Figure.push\_back(P);

}

if (RBVecOP->Checked) //Сторона Р

{

flpr=false;

if (ter==false)

{

TPoint P(mX,mY);

for (int i = 0; i<Figure.size();i++)

{

for (int rX = mX-10; rX<mX+10; rX++)

{

for (int rY = mY-10; rY<mY+10; rY++)

{

if ((rX == Figure[i].x)&&(rY == Figure[i].y))

{

LVecOP->Caption = "Точка 1: (" +

FloatToStrF(S2Wx(rX), ffFixed, 10, 2)+ "; " +

FloatToStrF(S2Wy(rY), ffFixed, 10, 2) +")";

TopF = i;

Tck.x=Figure[i].x;

Tck.y=Figure[i].y;

ter=true;

return;

}

}

}

}

ShowMessage("Выберите точку точнее!");

}

if (ter==true)

{

TPoint P1(mX,mY);

for (int i = 0; i<Figure.size();i++)

{

for (int rX = mX-10; rX<mX+10; rX++)

{

for (int rY = mY-10; rY<mY+10; rY++)

{

if ((rX == Figure[i].x)&&(rY == Figure[i].y))

{

Label9->Caption = "Точка 2: (" +

FloatToStrF(S2Wx(rX), ffFixed, 10, 2)+ "; " +

FloatToStrF(S2Wy(rY), ffFixed, 10, 2) +")";

TopF2 = i;

if ((TopF+1!=TopF2)&&(TopF!=TopF2+1)&&(TopF+Figure.size()-1!=TopF2)&&(TopF!=TopF2+Figure.size()-1))

{

TopF=0;

TopF2=0;

ShowMessage("Вы выбрали несоединенные точки");

proverka=true;

}

Tck2.x=Figure[i].x;

Tck2.y=Figure[i].y;

ter=false;

if (proverka==false)

{

printpr=false;

return;

}

if (proverka==true)

{

proverka=false;

printpr=true;

break;

}

}

}

}

}

ShowMessage("Выберите точку точнее!");

}

}

if (RBReflection->Checked) //точка для прямой

{

TPoint P3(mX,mY);

Prymaya.x=P3.x;

Prymaya.y=P3.y;

LReflection->Caption = "Точка на плоскости: (" +

FloatToStrF(S2Wx(Prymaya.x), ffFixed, 10, 2)+ "; " +

FloatToStrF(S2Wy(Prymaya.y), ffFixed, 10, 2) +")";

flpr=true;

}

if (RBTurn->Checked) //точка Е

{

flpr=false;

Tck = TPoint(mX,mY);

//строка с коодинатами точки Е

LTurn->Caption = "Точка поворота: (" + FloatToStrF(S2Wx(mX), ffFixed, 10, 2)

+ "; " + FloatToStrF(S2Wy(mY), ffFixed, 10, 2) +")";

}

Panel1->Repaint();

}

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

//угол поворота

void \_\_fastcall TForm1::ETurnChange(TObject \*Sender)

{

if (ETurn->Text != "")

{

Turn = StrToFloat(ETurn->Text);

Panel1->Repaint();

}

}

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

void \_\_fastcall TForm1::BCalcClick(TObject \*Sender)

{

//расчёт по матрице

if (RBMatrix->Checked)

{

//ввод из таблицы

Matrix[0][0] = StrToFloat(StringGrid->Cells[0][0]);

Matrix[0][1] = StrToFloat(StringGrid->Cells[1][0]);

Matrix[0][2] = StrToFloat(StringGrid->Cells[2][0]);

Matrix[1][0] = StrToFloat(StringGrid->Cells[0][1]);

Matrix[1][1] = StrToFloat(StringGrid->Cells[1][1]);

Matrix[1][2] = StrToFloat(StringGrid->Cells[2][1]);

Matrix[2][0] = StrToFloat(StringGrid->Cells[0][2]);

Matrix[2][1] = StrToFloat(StringGrid->Cells[1][2]);

Matrix[2][2] = StrToFloat(StringGrid->Cells[2][2]);

//расчёт

ReWFigure = DinWFigure;

for (int i = 0; i<DinWFigure.size(); i++)

{

double PM[3] = {DinWFigure[i].x, DinWFigure[i].y, 1.00};

double RM[3];

RM[0] = PM[0] \* Matrix[0][0] + PM[1] \* Matrix[1][0] + PM[2] \* Matrix[2][0];

RM[1] = PM[0] \* Matrix[0][1] + PM[1] \* Matrix[1][1] + PM[2] \* Matrix[2][1];

RM[2] = PM[0] \* Matrix[0][2] + PM[1] \* Matrix[1][2] + PM[2] \* Matrix[2][2];

ReWFigure[i].x = RM[0];

ReWFigure[i].y = RM[1];

}

FlagPaint = true;

BInputMClick(Sender);

W2Sf(ReFigure, ReWFigure);

}

}

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

//вывод матрицы

void \_\_fastcall TForm1::BInputMClick(TObject \*Sender)

{

StringGrid->Cells[0][0] = FloatToStrF(Matrix[0][0], ffFixed, 10, 2);

StringGrid->Cells[1][0] = FloatToStrF(Matrix[0][1], ffFixed, 10, 2);

StringGrid->Cells[2][0] = FloatToStrF(Matrix[0][2], ffFixed, 10, 2);

StringGrid->Cells[0][1] = FloatToStrF(Matrix[1][0], ffFixed, 10, 2);

StringGrid->Cells[1][1] = FloatToStrF(Matrix[1][1], ffFixed, 10, 2);

StringGrid->Cells[2][1] = FloatToStrF(Matrix[1][2], ffFixed, 10, 2);

StringGrid->Cells[0][2] = FloatToStrF(Matrix[2][0], ffFixed, 10, 2);

StringGrid->Cells[1][2] = FloatToStrF(Matrix[2][1], ffFixed, 10, 2);

StringGrid->Cells[2][2] = FloatToStrF(Matrix[2][2], ffFixed, 10, 2);

}

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

//выбор первой точки и второй

void \_\_fastcall TForm1::RBVecOPClick(TObject \*Sender)

{

LVecOP->Caption = "Точка 1: (" + FloatToStrF(S2Wx(Tck.x), ffFixed, 10, 2) + "; " + FloatToStrF(S2Wy(Tck.y), ffFixed, 10, 2) +")";

Label9->Caption= "Точка 2: ("+ FloatToStrF(S2Wx(Tck2.x), ffFixed, 10, 2) + "; "+ FloatToStrF(S2Wx(Tck2.y), ffFixed, 10, 2)+")";

}

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

//выбор первой точки Е

void \_\_fastcall TForm1::RBTurnClick(TObject \*Sender)

{

LTurn->Caption = "Точка E: (" + FloatToStrF(S2Wx(Tck.x), ffFixed, 10, 2)

+ "; " + FloatToStrF(S2Wy(Tck.y), ffFixed, 10, 2) +")";

}

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

//стирание фигуры

void \_\_fastcall TForm1::Button1Click(TObject \*Sender)

{

Figure.clear();

ReFigure.clear();

RBInput->Checked = true;

Matrix[0][0] = 1.00; Matrix[0][1] = 0.00; Matrix[0][2] = 0.00;

Matrix[1][0] = 0.00; Matrix[1][1] = 1.00; Matrix[1][2] = 0.00;

Matrix[2][0] = 0.00; Matrix[2][1] = 0.00; Matrix[2][2] = 1.00;

BInputMClick(Sender);

}

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

//динамика

void \_\_fastcall TForm1::Timer1Timer(TObject \*Sender)

{

DinWFigure = ReWFigure;

StartPaint = true;

Panel1->Repaint();

if (SpeedButton1->Down) BCalcClick(Sender);

}

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

//сброс после динамики

void \_\_fastcall TForm1::Button2Click(TObject \*Sender)

{

StartPaint = false;

PaintEnd = false;

FlagPaint = false;

//DinFigure = Figure;

S2Wf(Figure, DinWFigure);

if (RBMatrix->Checked) ReFigure = Figure;

Panel1->Repaint();

}

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

void \_\_fastcall TForm1::Button3Click(TObject \*Sender)

{

if ((EOXrate->Text == "0,")||(EOXrate->Text == "0,0")||(EOXrate->Text == "0,00")||(EOXrate->Text == "0,000"))

return;

if (EOXrate->Text != "")

{

OXrate = StrToFloat(EOXrate->Text);

if (OXrate==0)

{

ShowMessage("Недопустимое значение!"); OXrate=0.1;

EOXrate->Text = "0,1"; return;

}

Panel1->Repaint();

}

}

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

5 Результаты работы программы

Результаты работы программы представлены на рисунках 4 - 12.

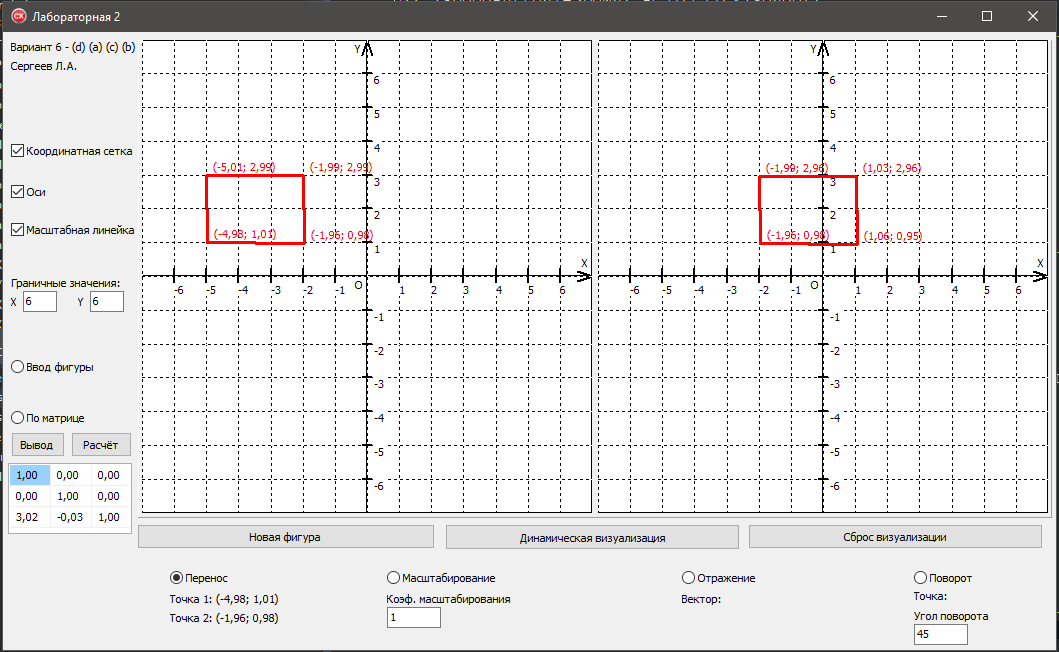


Рисунок 4 – Перенос вдоль стороны Aв(-4,98;1,01) Bв(-1,96;0,98)

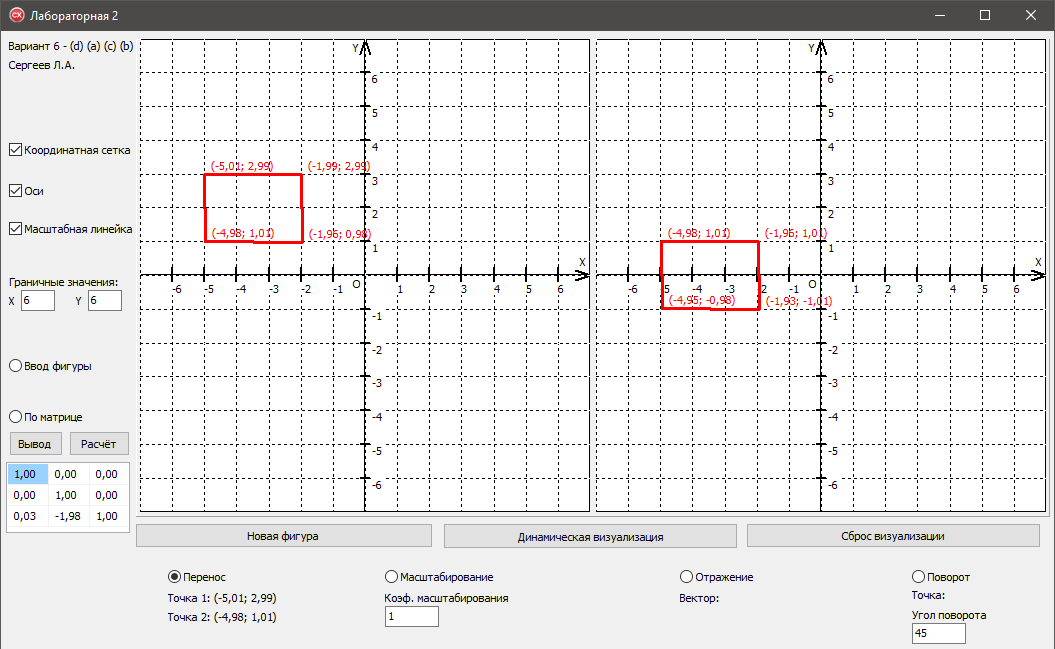


Рисунок 5- Перенос вдоль стороны Aв(-5,02;2,99) Bв(-4,98;1,01)

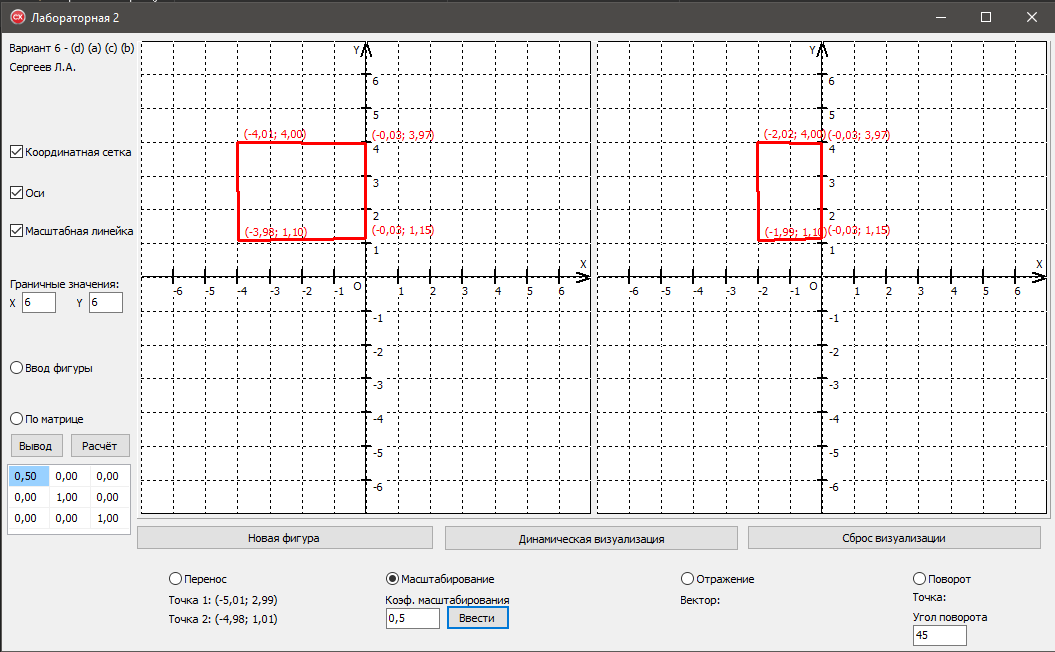


Рисунок 6 - Масштабирование вдоль оси Ox c кф 0,5

## 

Рисунок 7 - Масштабирование вдоль оси Ox c кф 1,25

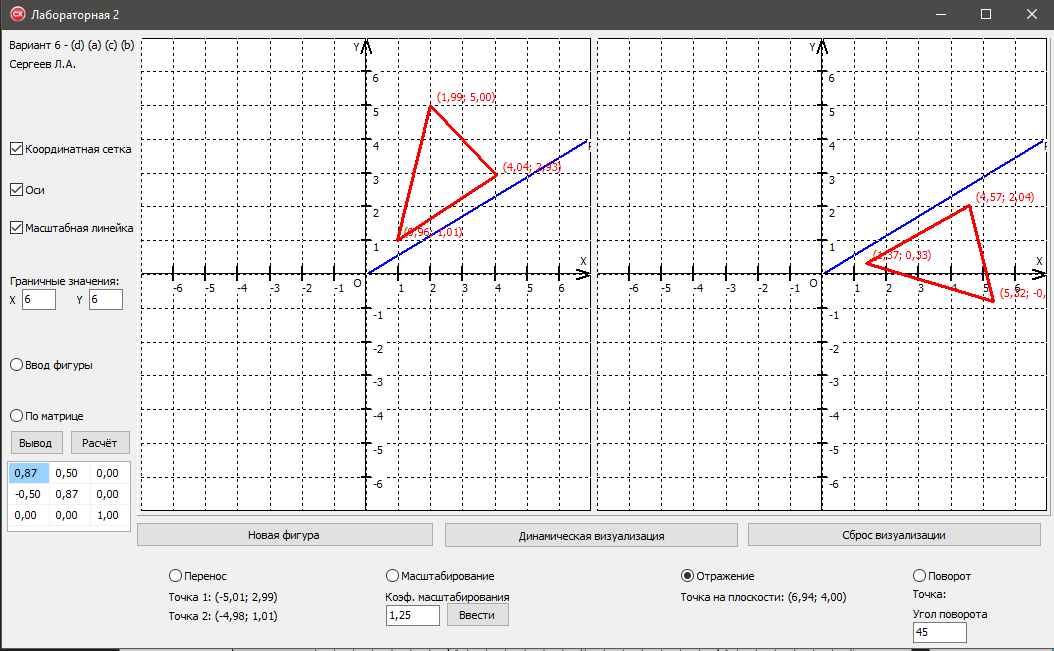


Рисунок 8 – Отражение относительно вектора OP

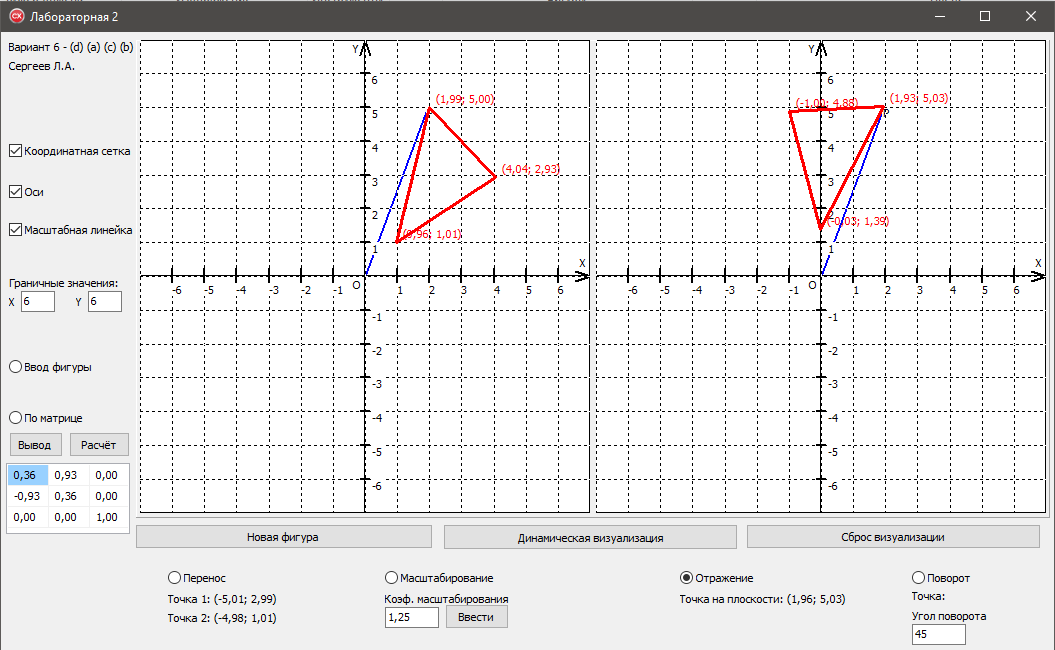


Рисунок 9 - Отражение относительно вектора OP со всходом в вершину треугольника

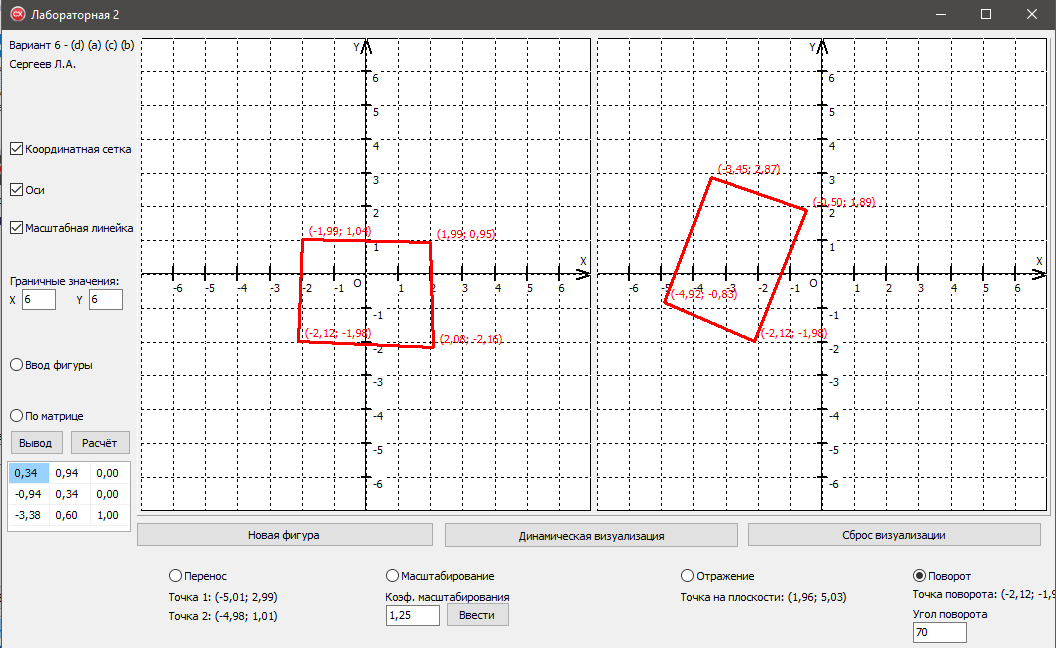


Рисунок 10 – Поворот относительно вершины (-2,12;-1,98) и угол поворота равен 70 градусам

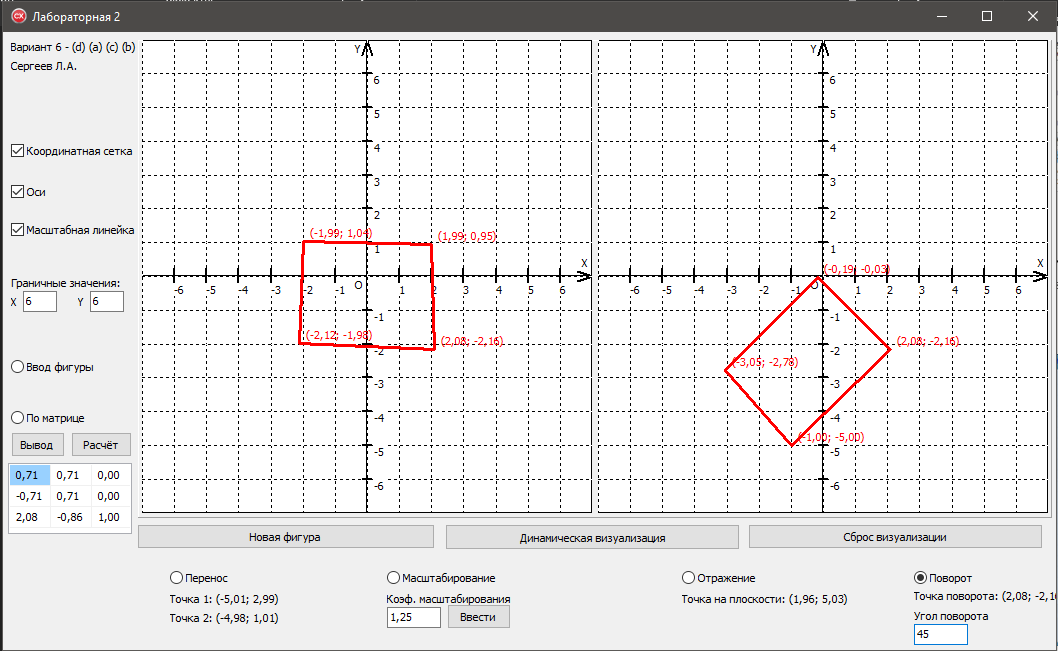


Рисунок 11 - Поворот относительно вершины (2,08;-2,16) и угол поворота равен 45 градусов

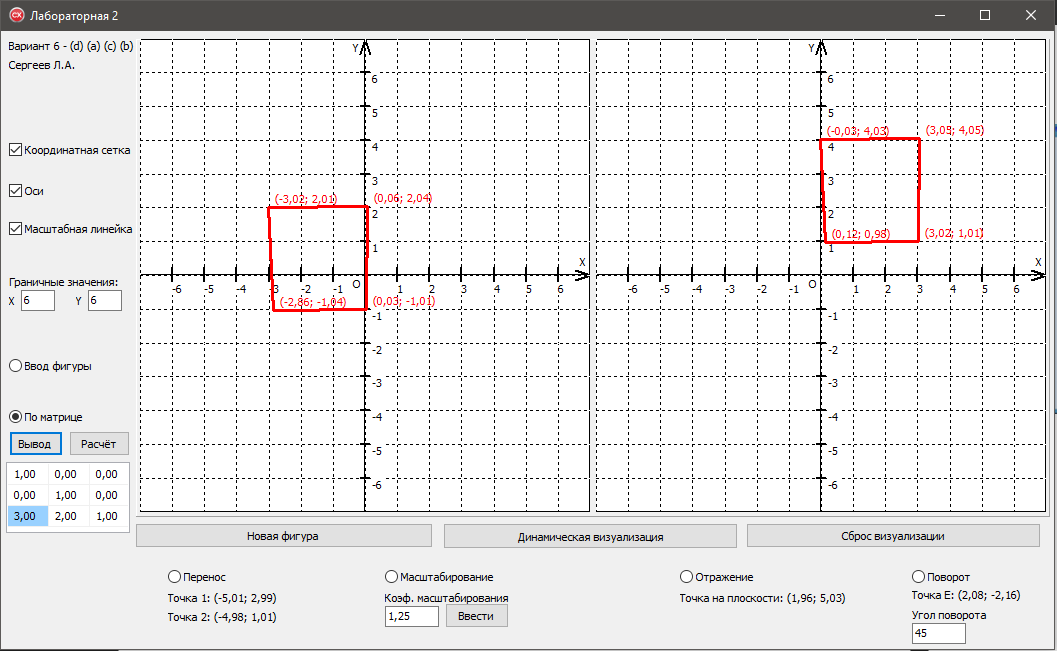


Рисунок 12 – Применение преобразованной матрицы к исходной фигуре

6 Выводы

В результате выполнения лабораторной работы я изучил математические методы аффинных преобразований на плоскости и практически освоил прием программной реализации аффинных преобразований.