Министерство образования и науки Российской Федерации Федеральное агентство по образованию Федеральное государственное бюджетное образовательное учреждение высшего образования «Вятский государственный университет»

Факультет автоматики и вычислительной техники

Кафедра электронных вычислительных машин

Лабораторная работа №3

по курсу «Компьютерная графика»

# «Аффинные преобразования на плоскости»

Выполнил студент группы ИВТ-21\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Птахова А.М/

Проверил \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ Коржавина А.С./

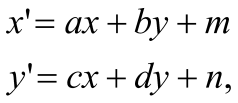
Киров 2021

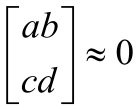
**Цель работы**: закрепить лекционный материал по изучению одноименной темы, реализовав матрицы переноса, масштабирования, отражения и вращения применительно к координатам описанной в программе плоской фигуры (многоугольника) с целью демонстрации движения и преобразования формы этой фигуры на плоскости.

**Краткие теоретические сведения:**

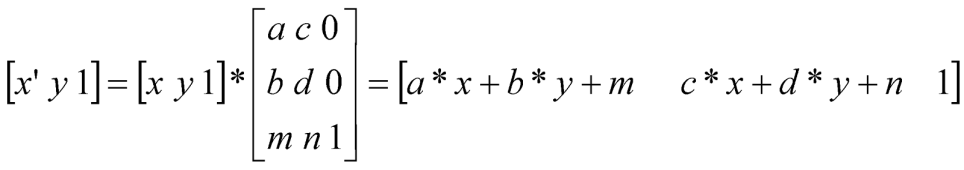
При решении задач компьютерной графики однородные координаты обычно вводятся так: произвольной точке М(x,y) на плоскости ставится в соответствие точка M(x,y,1) в пространстве.

Если нам необходимо преобразовать точку на плоскости с координатами (x,y) в другую точку то задача сводится к поиску новых координат для этой точки - (x). В случае аффинных преобразований такой поиск сведется к решению уравнений

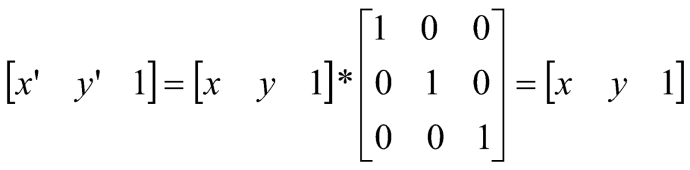
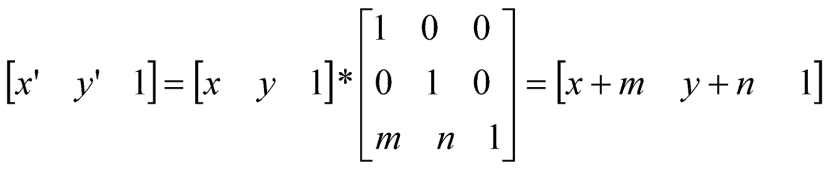
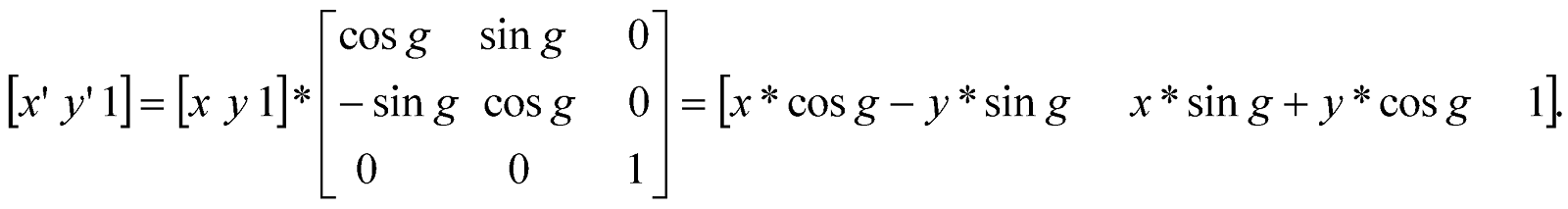


где a, b, c, d, m, n - произвольные числа, причем:.

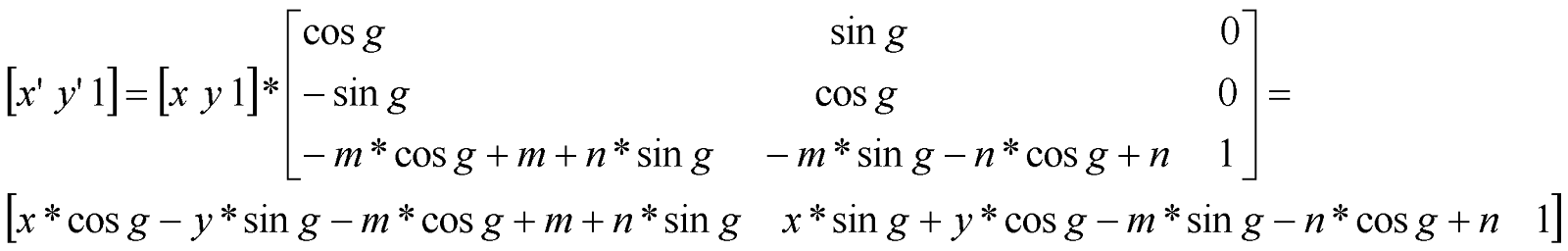
В случае, когда m и n не равны нулю, для представления преобразования в матричной форме нужно исходные и преобразованные координаты точки записать в однородных координатах (x,y,1) и (https://lh5.googleusercontent.com/IsQQNWL_UVaFyiWySn1xM1yiYFQH7mahmDKo-zfpbj9ivuZ7bI_aYAomZkDKFC9ZDi3QFru8lGU7OKDUi8R5KFFkrosXHPDVffh0cWLOdpbafM7bSK6WfGR7lpZaByDTDrz6cVE=s1600). Тогда в матричной форме общий вид преобразования будет следующим



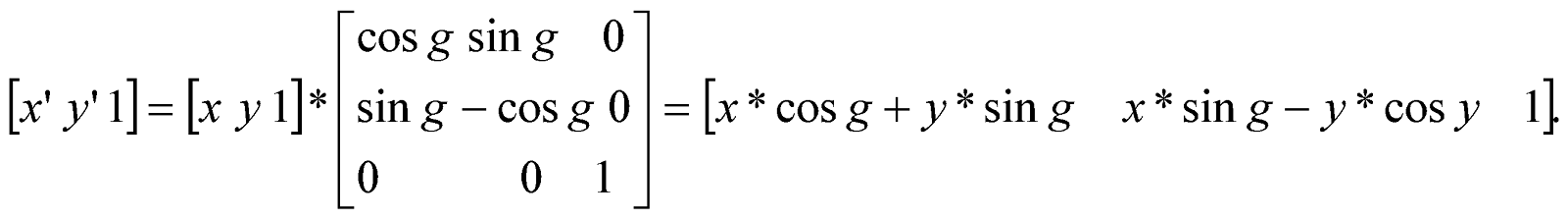
Наибольшее распространение получили частные случаи аффинных преобразований:

1. Единичное преобразование. Единичная матрица оставляет точку неподвижной. 
2. Сдвиг или плоско-параллельный перенос. Матрица переводит точку на m единиц вдоль оси x и на n -вдоль оси y:.
3. Вращение вокруг начала координат. Его матрица осуществляет поворот точки объекта на угол g против часовой стрелки: 
4. Вращение вокруг произвольного центра осуществляет поворот вокруг точки (m,n) на угол g против часовой стрелки. Преобразование выполняется как последовательность трех элементарных:
   * сдвиг центра вращения (m,n) в начало координат с помощью матрицы сдвига
   * поворот на угол g вокруг начала координат с помощью матрицы вращения
   * сдвиг точки (m,n) в исходное положение, используя матрицу сдвига

Итоговое преобразование будет выглядеть так:

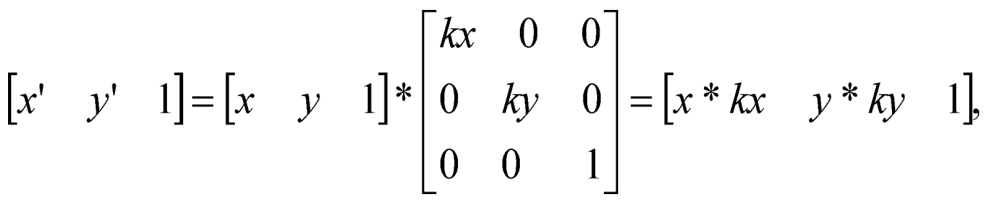


5.Симметрия относительно оси, проходящей через начало координат осуществляется преобразованием вида



6.Масштабирование – увеличение (уменьшение размеров изображения) - в общем случае изменяет форму объекта. Назначается точка, относительно которой производится преобразование.

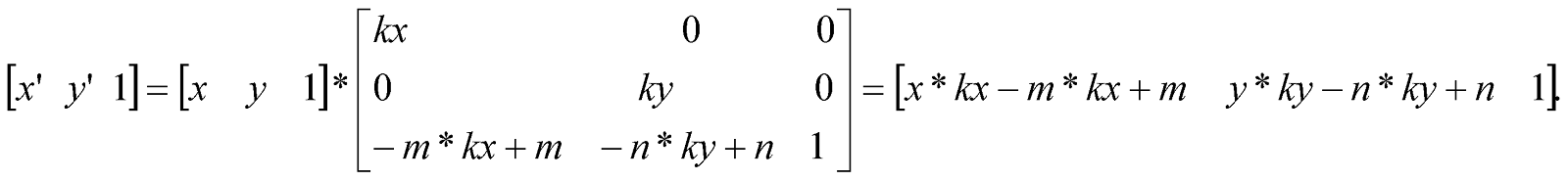
Масштабирование относительно точки О - начала координат



где kx ,ky - коэффициенты искажения по осям Ox, Oy соответственно.

При kx=ky=k осуществляется преобразование подобия, при kx # ky изображение искажается. Изображение увеличивается при k > 1 и уменьшается при k < 1.

Масштабирование относительно произвольной точки с координатами (m, n)



Матрица любого аффинного преобразования может быть получена умножением соответствующих рассмотренных здесь простых матриц. Порядок умножения имеет значение, поэтому выполнять его надо в определенной логической последовательности.

**Разработка программы**

Пользователь может перемещать, вращатъ и масштабировать фигуру при помощи соответствующих кнопок.

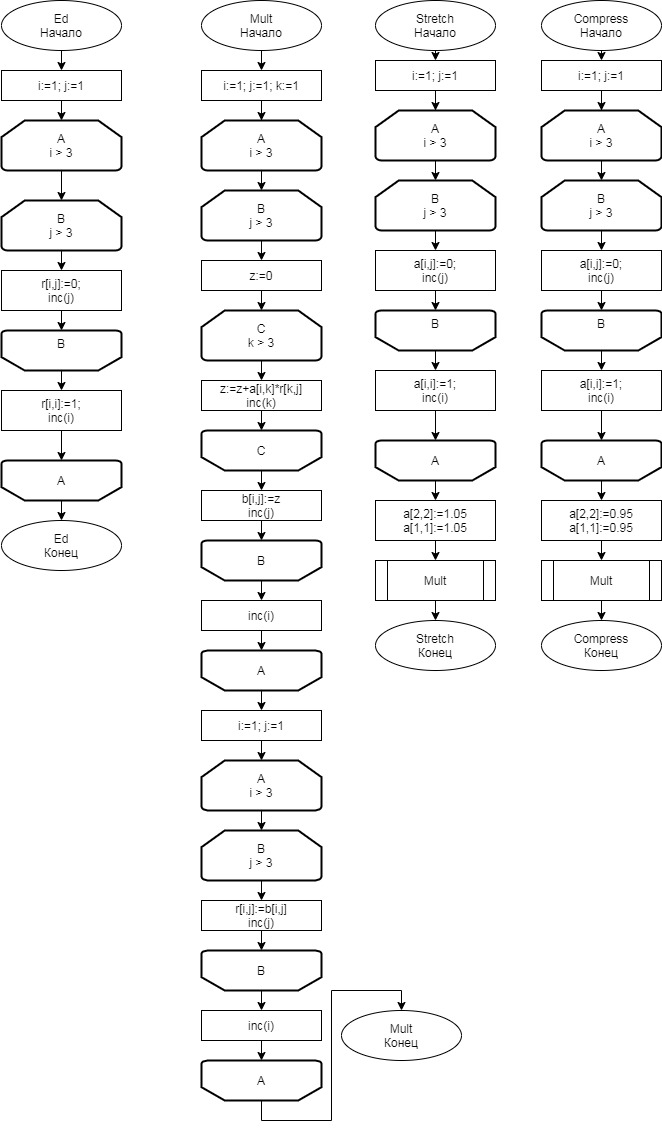
Схемы алгоритмов приведены в приложении А, листинг программ приведет в приложении Б, экранные формы – в приложении В

**Вывод:**

В ходе выполнения лабораторной работы были получены знания об аффинных преобразованиях на плоскости в общем виде и частных случаях. В результате была написана программа, реализующая все изученные методы.

Приложение А

Схемы алгоритмов



Приложение Б

Листинг программы

**procedure** Ed; { присвоение матрице R значения единичной }

**var** i,j:integer;

**begin**

**for** i:=1 **to** n **do**

**begin** { 1 0 0 }

**for** j:=1 **to** n **do** r[i,j]:=0; { 0 1 0 }

r[i,i]:=1; { 0 0 1 }

**end**;

**end**;

**procedure** Mult; {умножение матриц А и R: R = B = A\*R }

**var** b:mtr;

z:real;

i,j,k:integer;

**begin**

**for** i:=1 **to** n **do**

**for** j:=1 **to** n **do**

**begin**

z:=0;

**for** k:=1 **to** n **do**

z:=z+a[i,k]\*r[k,j];

b[i,j]:=z

**end**;

**for** i:=1 **to** n **do**

**for** j:=1 **to** n **do**

r[i,j]:=b[i,j]

**end**;

**procedure** Stretch; {расчет матриц А и R для растягивания фигуры}

**var** i,j:integer;

**begin**

**for** i:=1 **to** n **do**

**begin** { 1 0 0 }

**for** j:=1 **to** n **do** a[i,j]:=0; { 0 1.05 0 }

a[i,i]:=1; { 0 0 1 }

**end**;

a[2,2]:=1.05;{коэффициент растяжения}

a[1,1]:=1.05;

Mult;

**end**;

**procedure** Compress; {расчет матриц А и R для сжатия фигуры}

**var** i,j:integer;

**begin**

**for** i:=1 **to** n **do**

**begin** { 1 0 0 }

**for** j:=1 **to** n **do** a[i,j]:=0; { 0 1.05 0 }

a[i,i]:=1; { 0 0 1 }

**end**;

a[2,2]:=0.95;{коэффициент растяжения}

a[1,1]:=0.95;

Mult;

**end**;

**procedure** MirrorX; {расчет матриц А и R для отражения фигуры по Х}

**var** i,j:integer;

**begin**

**for** i:=1 **to** n **do**

**begin** { 1 0 0 }

**for** j:=1 **to** n **do** a[i,j]:=0; { 0 -1 0 }

a[i,i]:=1; { 0 0 1 }

**end**;

a[2,2]:=-1;

Mult;

**end**;

**procedure** MirrorY; {расчет матриц А и R для отражения фигуры по Х}

**var** i,j:integer;

**begin**

**for** i:=1 **to** n **do**

**begin** { -1 0 0 }

**for** j:=1 **to** n **do** a[i,j]:=0; { 0 1 0 }

a[i,i]:=1; { 0 0 1 }

**end**;

a[1,1]:=-1;

Mult;

**end**;

**procedure** MoveY(dy:integer); {расчет матриц А и R для перемещения фигуры по Y}

**var** i,j:integer;

**begin**

**for** i:=1 **to** n **do**

**begin** { 1 0 0 }

**for** j:=1 **to** n **do** a[i,j]:=0; { 0 1 dy }

a[i,i]:=1; { 0 0 1 }

**end**;

a[2,3]:=dy;

Mult;

**end**;

**procedure** MoveX(dx:integer); {расчет матриц А и R для перемещения фигуры по Y}

**var** i,j:integer;

**begin**

**for** i:=1 **to** n **do**

**begin** { 1 0 dx }

**for** j:=1 **to** n **do** a[i,j]:=0; { 0 1 0 }

a[i,i]:=1; { 0 0 1 }

**end**;

a[1,3]:=dx;

Mult;

**end**;

**procedure** Rotate(u:real); {расчет матриц А и R для поворота фигуры}

**var** c, s: real; {---н угол alfa(р д)---}

i,j:integer;

**begin** { cos(u) -sin(u) 0 }

**for** i:=1 **to** 3 **do** { sin(u) cos(u) 0 }

**for** j:=1 **to** 3 **do** { 0 0 1 }

a[i,j]:=0;

a[3,3]:=1;

c:=cos(u); a[1,1]:= c; a[2,2]:=c;

s:=sin(u); a[1,2]:=-s; a[2,1]:=s;

Mult;

**end**;

**procedure** New\_XY;{расчет новых координат фигуры по исходным}

**var** i,j:integer;

**begin** {с использованием матрицы преобразования R}

**for** i:=1 **to** m **do**

**begin**

xb[i]:=xa[i]\*r[1, 1]+ ya[i]\*r[1, 2]+ r[1, 3];

yb[i]:=xa[i]\*r[2, 1]+ ya[i]\*r[2, 2]+ r[2, 3]

**end**;

**end**;

Приложение В

Экранные формы

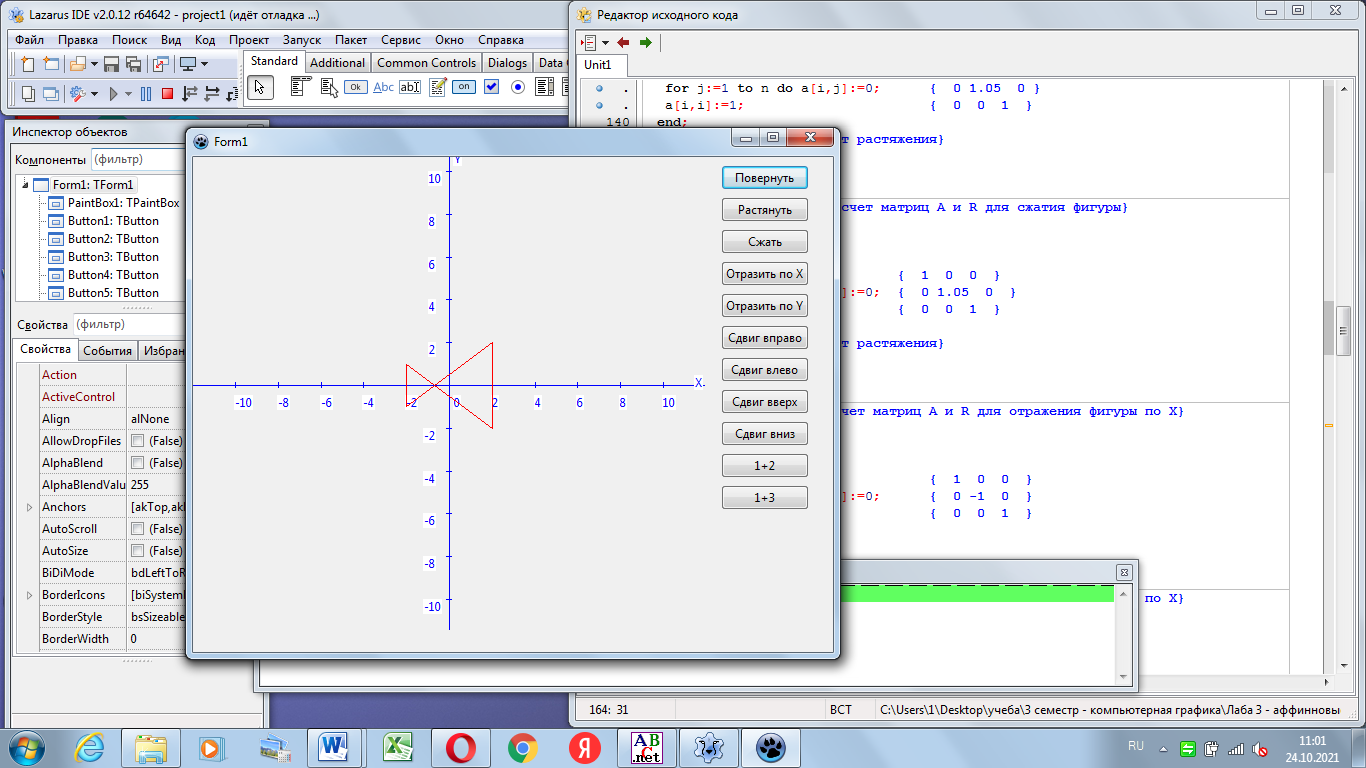


Рисунок 1 – начальное расположение фигуры

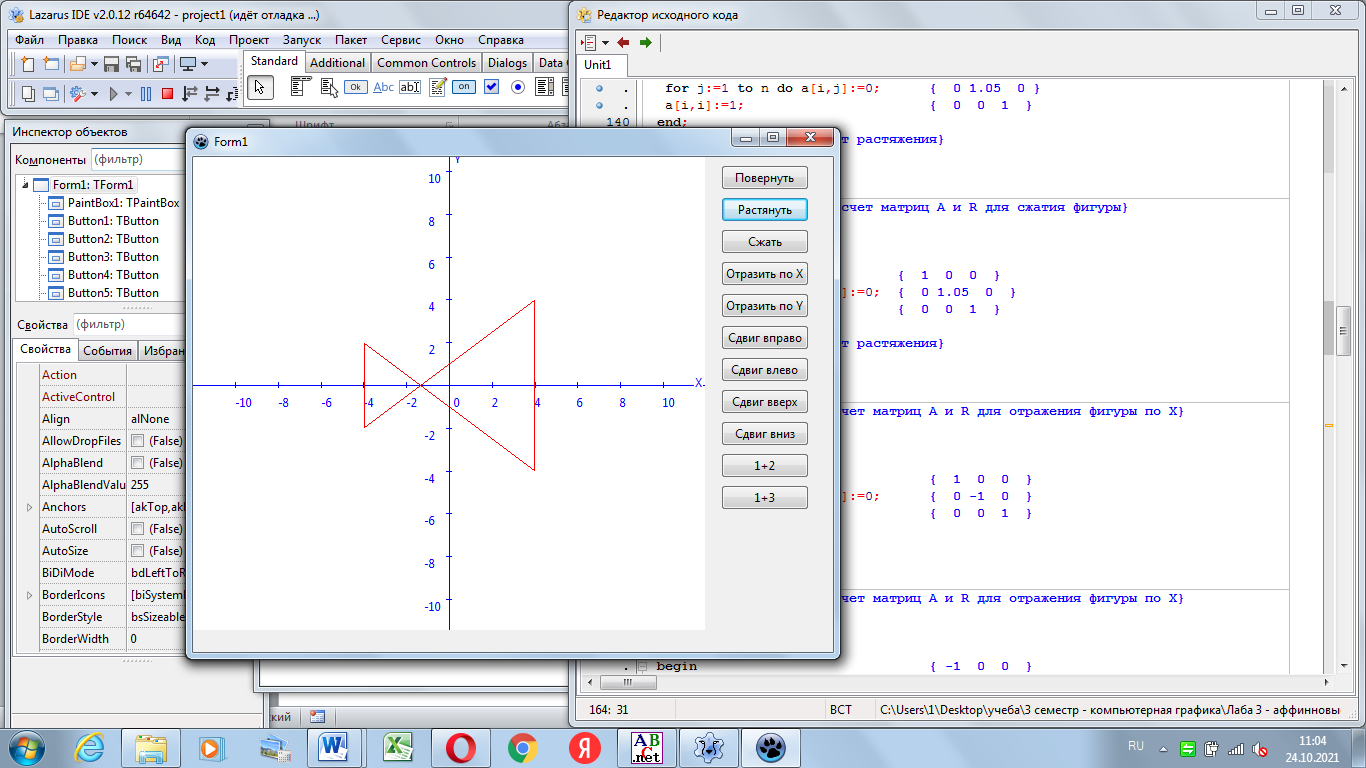


Рисунок 2 – растяжение фигуры

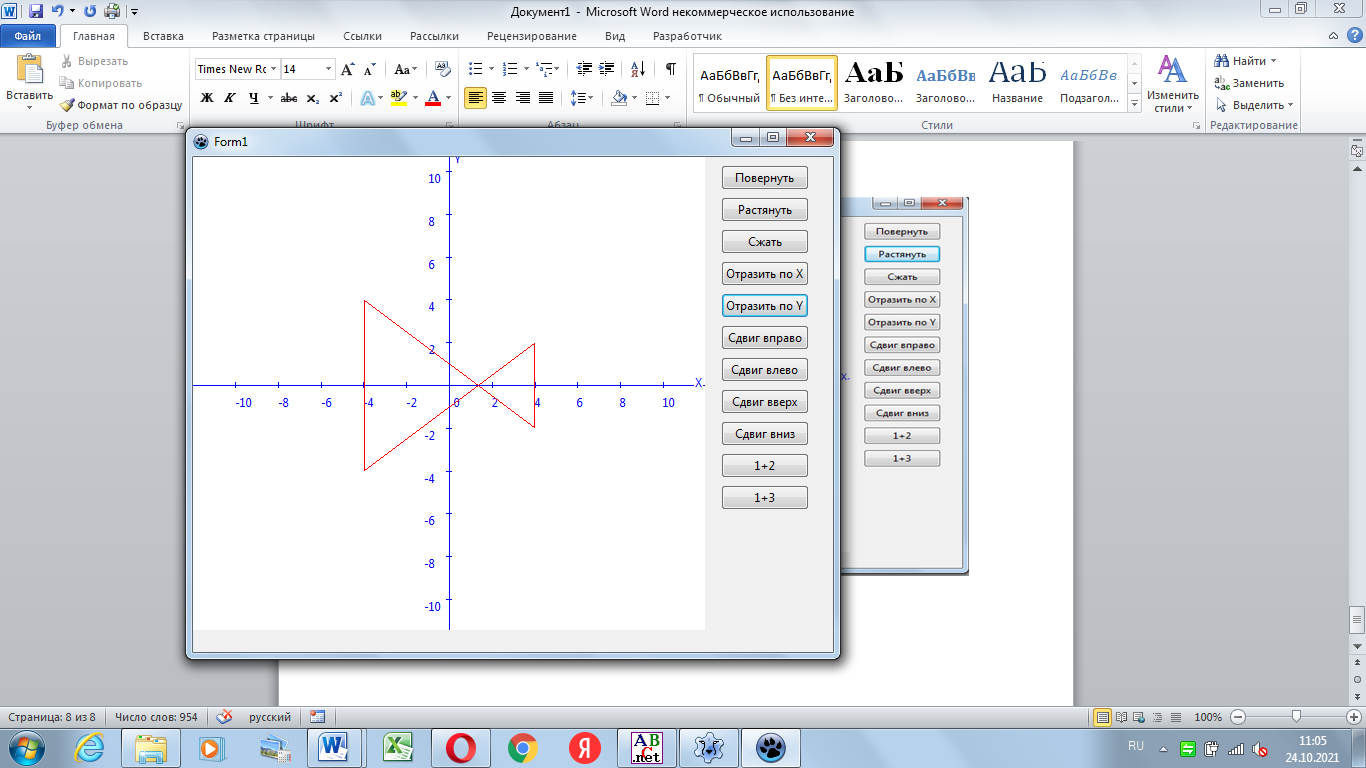


Рисунок 3 – отражение фигуры по Y