Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ
УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)**
Кафедра автоматизации обработки информации (АОИ)

2D аффинные преобразования

Отчет о выполнении лабораторной работы
по дисциплине «Компьютерная графика»

Студент гр. 429-3
_____ Бабец А. А.
«___» _____ 2021 г.

Руководитель
канд. техн. наук., доцент каф. АОИ
_____ Т. О. Перемитина
«___» _____ 2021 г.

Томск 2021

Введение

Цель: применение аффинных преобразований в двумерном пространстве.

Постановка задачи:

- построить двумерное изображение заданной фигуры (начальное положение фигуры).
- выполнить аффинные преобразования (поворот, масштабирование, отражение и сдвиг) – путем реализации процедуры (функции) умножения матрицы начальных координат фигуры на матрицу преобразований.
- отобразить новое положение фигуры.

Вид 2D фигуры в соответствии с вариантом № 2.

2 Содержание

2.1 Используемая среды программирования

Рассмотрим особенности среды программирования Microsoft Visual Studio:

- подсветка синтаксиса и простое автозавершение кода;
- анализ кода при загрузке и непосредственно при вводе;
- понятный и удобный интерфейс программы позволяет легко и быстро привыкнуть к работе и повышает её продуктивность;
- оснащенность инструментами для сборки, средой выполнения тестов, инструментами покрытия и встроенным терминальным окном.

Вышеперечисленные особенности послужили тому, что была выбрана среда разработки Microsoft Visual Studio для написания кода.

2.2 Метод решения задачи

Все преобразования можно выполнить с помощью четырех базовых операций: переноса (сдвиг); масштабирования (увеличения или уменьшения размеров); отражения и вращения изображения. Двумерные фигуры представляются в виде трехмерной матрицы с использованием однородных координат.

I. Матрица вращения:

$$[R] = \begin{bmatrix} \cos\varphi & \sin\varphi & 0 \\ -\sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

II. Матрица масштабирования:

$$[D] = \begin{bmatrix} \alpha & 0 & 0 \\ 0 & \delta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

III. Матрица отражения:

$$[M] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

IV. Матрица переноса:

$$[T] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \lambda & \mu & 1 \end{bmatrix}$$

Преобразования производятся умножением матриц преобразований на матрицу вершин фигуры и присваиванием новых значений последним. Таким образом, преобразования выполняются над множеством вершин фигуры, после чего результат преобразований отображается с новыми координатами.

2.3 Программа

Листинг программы представлен в приложении А.

Основные методы взаимодействия с 2D объектом в аффинном пространстве представлены в основном классе Form. При нажатии на каждую кнопку происходит составление соответствующей матрицы и передача её в метод Multiply, производящий умножение матриц.

Листинг Multiply:

```
public double[,] Multiply(double[,] A, double[,] B)
        {
            double[,] C = new double[A.GetLength(0), B.GetLength(1)];
            for (int i = 0; i < C.GetLength(0); i++)
            {
                int z = 0;
                for (int k = 0; k < C.GetLength(1); k++)
                {
                    double sum = 0;
                    for (int j = 0; j < C.GetLength(1); j++)
                    {
                        sum += A[i, j] * B[j, k];
                    }
                    C[i, z] = sum;
                    z++;
                }

            }
            return C;
        }
```

2.4 Функционал

Программа представляет собой Windows форму (рисунок 2.4.1). Основная часть программы отображает координатные оси и фигуру.

В левой части программы расположено меню, представленное пятью пунктами: сброс, перенос, вращение, масштабирование и отражение.

Первый пункт включат в себя только кнопку «Reset», приводящую фигуру в изначальное положение.

Во втором и четвёртом пунктах нужно вводить параметры в поля «X» и «Y», нажав на кнопку выше («Move» или «Scale»), фигура на экране изменится согласно соответствующему аффинному преобразованию.

Третий пункт включает в себя только одно поле ввода и кнопку («Rotate») для запуска процедуры изменения.

Пятый пункт содержит две кнопки для изменения фигуры – «Mirror X» и «Mirror Y».

Результаты работы программы приведены в приложении Б.

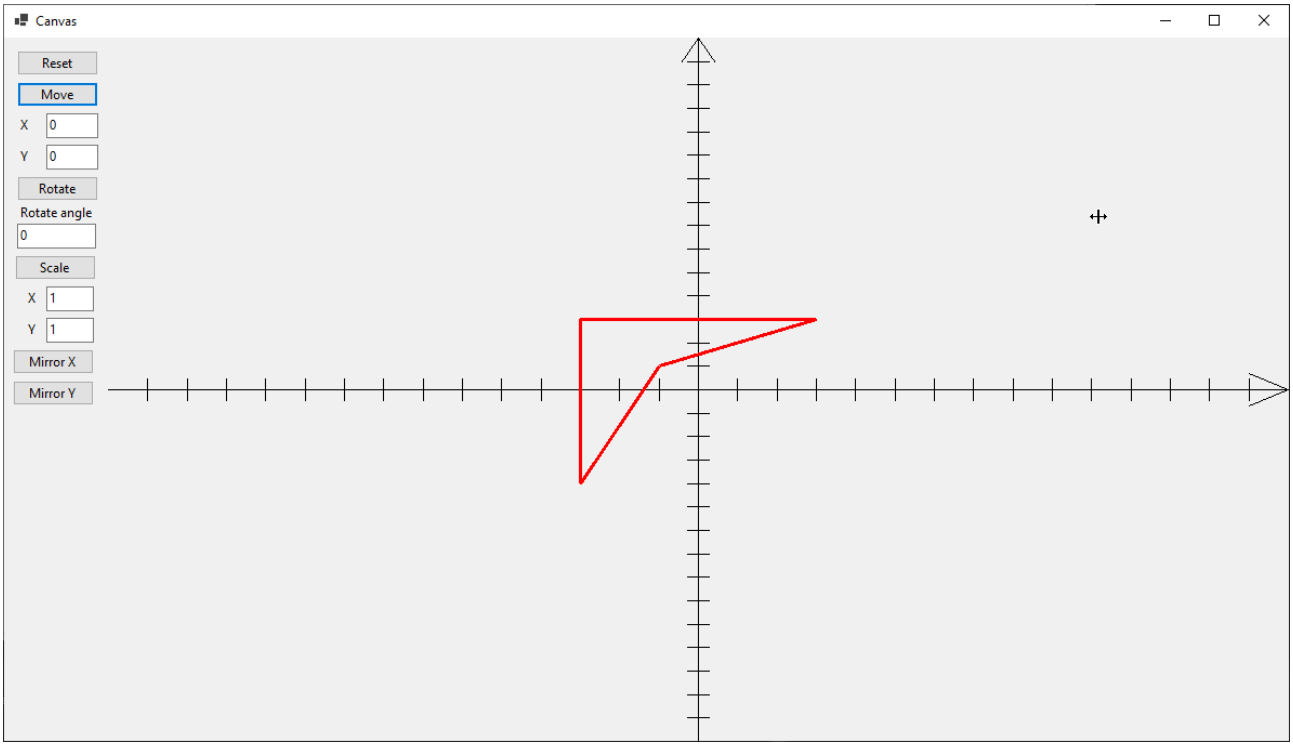


Рисунок 2.4.1 – GUI программы

Заключение

В ходе работы были изучены теоретические основы 2D аффинных преобразований, приобретены практические навыки построения 2D изображения, реализованы базовые методы взаимодействия с 2D объектом в аффинном пространстве.

Разработана программа, позволяющая пользователю проводить аффинные преобразования с 2D фигурой.

Приложение А

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace prakt
{
    Ссылок: 3
    public partial class Form1 : Form
    {
        Pen pen = new Pen(Color.Black, 1);
        float step_x, step_y;
        double[,] figure;
        ссылка: 1
        public Form1()
        {
            InitializeComponent();
            canvas.Paint += PictureBox_Paint;
        }
        ссылка: 1
        private void PictureBox_Paint(object sender, PaintEventArgs e)
        {
            Refresh();
            float Ox = canvas.Width / 2, Oy = canvas.Height / 2;
            step_x = Ox / 15;
            step_y = Oy / 15;
            figure = new double[,] {
                { -1 * step_x, 1 * step_y, 1 },
                { 3 * step_x, 3 * step_y, 1 },
                { -3 * step_x, 3 * step_y, 1 },
                { -3 * step_x, -4 * step_y, 1 }
            };

            //рисуем оси;
            e.Graphics.TranslateTransform(Ox, Oy);
            DrawAxises(e.Graphics, Ox, Oy);

```

Листинг Form1 - 1

```
            //рисуем фигуру
            DrawFigure(e.Graphics);

        }


        ссылка: 1
        public void DrawAxises(Graphics e, float ox, float oy)
        {
            pen = new Pen(Color.Black, 1);
            e.DrawLine(pen, 0, -oy, 0, oy);
            e.DrawLine(pen, -ox, 0, ox, 0);

            for (float x = 0; x < ox; x +=step_x)
            {
                e.DrawLine(pen, x, 10, x, -10);
                e.DrawLine(pen, -x, 10, -x, -10);
            }

            for (float y = 0; y < oy; y += step_y)
            {
                e.DrawLine(pen, -10, y, 10, y);
                e.DrawLine(pen, -10, -y, 10, -y);
            }

            e.DrawLine(pen, 0, -oy, 15, -oy + step_y);
            e.DrawLine(pen, -15, -oy + step_y, 0, -oy);

            e.DrawLine(pen, ox, 0, ox - step_x, 15);
            e.DrawLine(pen, ox, 0, ox - step_x, -15);
        }

```

Листинг Form1 - 2

```
    ссылка: 1
    private void Move_Click(object sender, EventArgs e)
    {
        double[,] move = {
            { 1, 0, 0 },
            { 0, 1, 0 },
            { int.Parse(x_move.Text)* step_x, int.Parse(y_move.Text)* step_y, 1 }
        };
        figure = Multiply(figure, move);
    }

    ссылка: 1
    private void reset_Click(object sender, EventArgs e)
    {
        figure = new double[,] {
            { -1 * step_x, 1 * step_y, 1 },
            { 3 * step_x, 3 * step_y, 1 },
            { -3 * step_x, 3 * step_y, 1 },
            { -3 * step_x, -4 * step_y, 1 }
        };
    }

    ссылка: 1
    public void DrawFigure(Graphics e)
    {
        pen = new Pen(Color.Red, 3);
        e.DrawLine(pen, Convert.ToInt32(figure[0,0]) , Convert.ToInt32(-figure[0,1]) , Convert.ToInt32(figure[1,0]) , Convert.ToInt32(-figure[1,1]) );
        e.DrawLine(pen, Convert.ToInt32(figure[1,0]) , Convert.ToInt32(-figure[1,1]) , Convert.ToInt32(figure[2,0]) , Convert.ToInt32(-figure[2,1]) );
        e.DrawLine(pen, Convert.ToInt32(figure[2,0]) , Convert.ToInt32(-figure[2,1]) , Convert.ToInt32(figure[3,0]) , Convert.ToInt32(-figure[3,1]) );
        e.DrawLine(pen, Convert.ToInt32(figure[3,0]) , Convert.ToInt32(-figure[3,1]) , Convert.ToInt32(figure[0,0]) , Convert.ToInt32(-figure[0,1]) );
    }
```

Листинг Form1 - 3

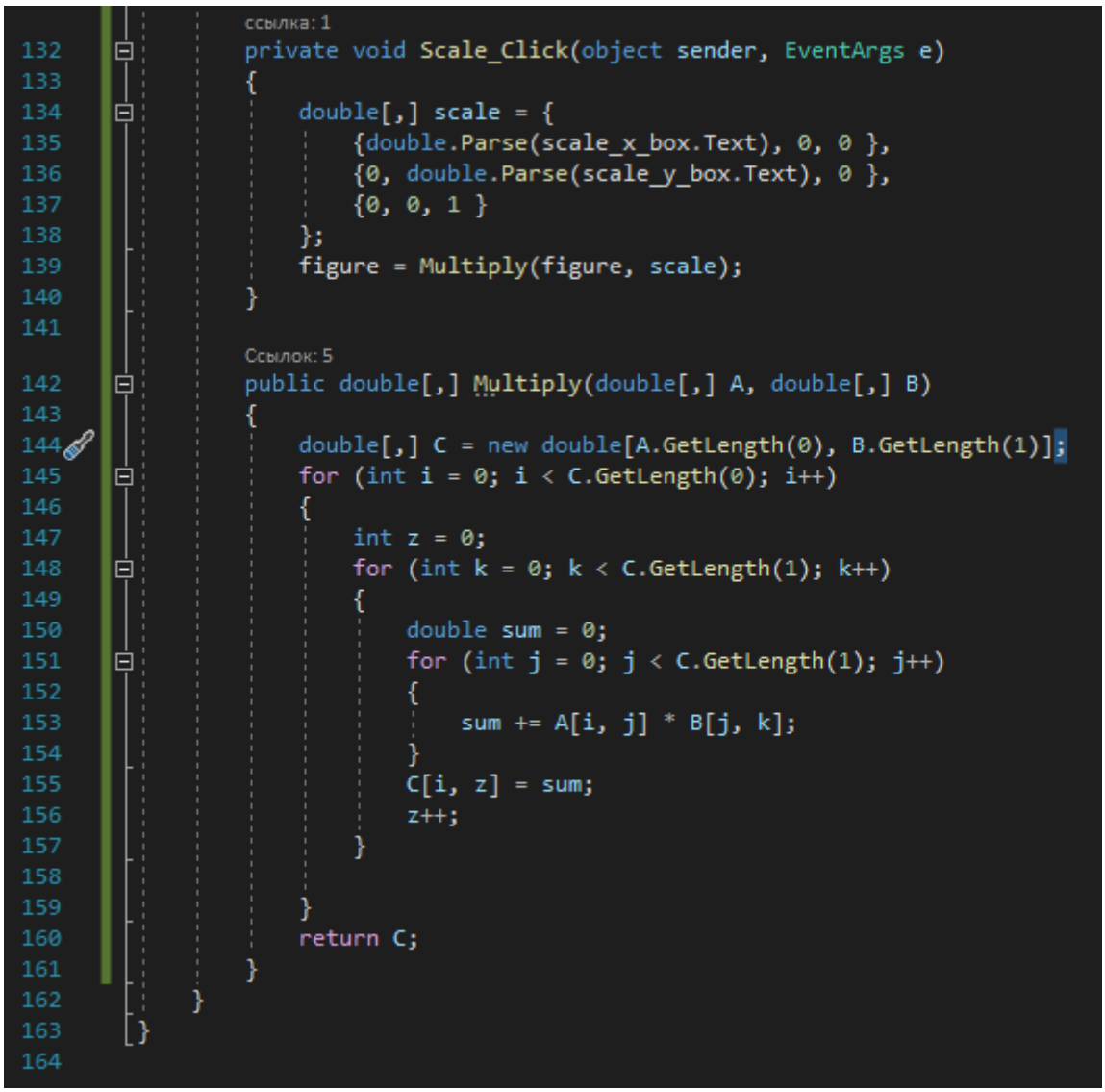
```
ссылка: 1
private void Rotate_Click(object sender, EventArgs e)
{
    double angle = double.Parse(rotate_angle.Text);
    double[,] rotate = {
        {Math.Cos(d: Math.PI / 180 * -angle), Math.Sin(Math.PI / 180 * -angle), 0},
        {-Math.Sin(Math.PI / 180 * -angle), Math.Cos(Math.PI / 180 * -angle), 0},
        { 0, 0, 1}
    };

    figure = Multiply(figure, rotate);
}

ссылка: 1
private void Mirror_X_Click(object sender, EventArgs e)
{
    double[,] mirror = {
        {-1, 0, 0 },
        {0, 1, 1 },
        {0, 0, 1 }
    };
    figure = Multiply(figure, mirror);
}

ссылка: 1
private void Mirror_Y_Click(object sender, EventArgs e)
{
    double[,] mirror = {
        {1, 0, 0 },
        {0, -1, 1 },
        {0, 0, 1 }
    };
    figure = Multiply(figure, mirror);
}
```

Листинг Form1 - 4



```
        ссылка: 1
132     private void Scale_Click(object sender, EventArgs e)
133     {
134         double[,] scale = {
135             {double.Parse(scale_x_box.Text), 0, 0 },
136             {0, double.Parse(scale_y_box.Text), 0 },
137             {0, 0, 1 }
138         };
139         figure = Multiply(figure, scale);
140     }
141
        Ссылок: 5
142     public double[,] Multiply(double[,] A, double[,] B)
143     {
144         double[,] C = new double[A.GetLength(0), B.GetLength(1)];
145         for (int i = 0; i < C.GetLength(0); i++)
146         {
147             int z = 0;
148             for (int k = 0; k < C.GetLength(1); k++)
149             {
150                 double sum = 0;
151                 for (int j = 0; j < C.GetLength(1); j++)
152                 {
153                     sum += A[i, j] * B[j, k];
154                 }
155                 C[i, z] = sum;
156                 z++;
157             }
158
159         }
160         return C;
161     }
162 }
163 }
164
```

Листинг Form1 - 5

Приложение Б

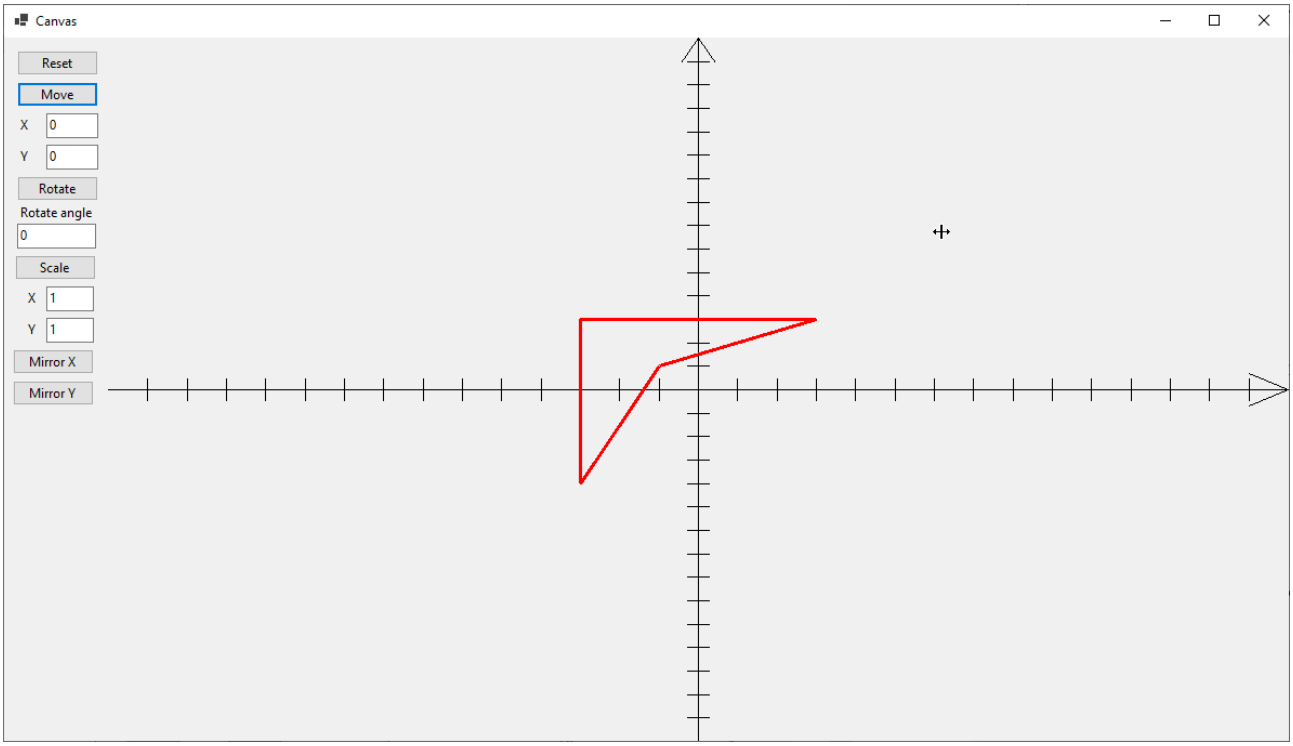


Рисунок 1 – Начальное состояние

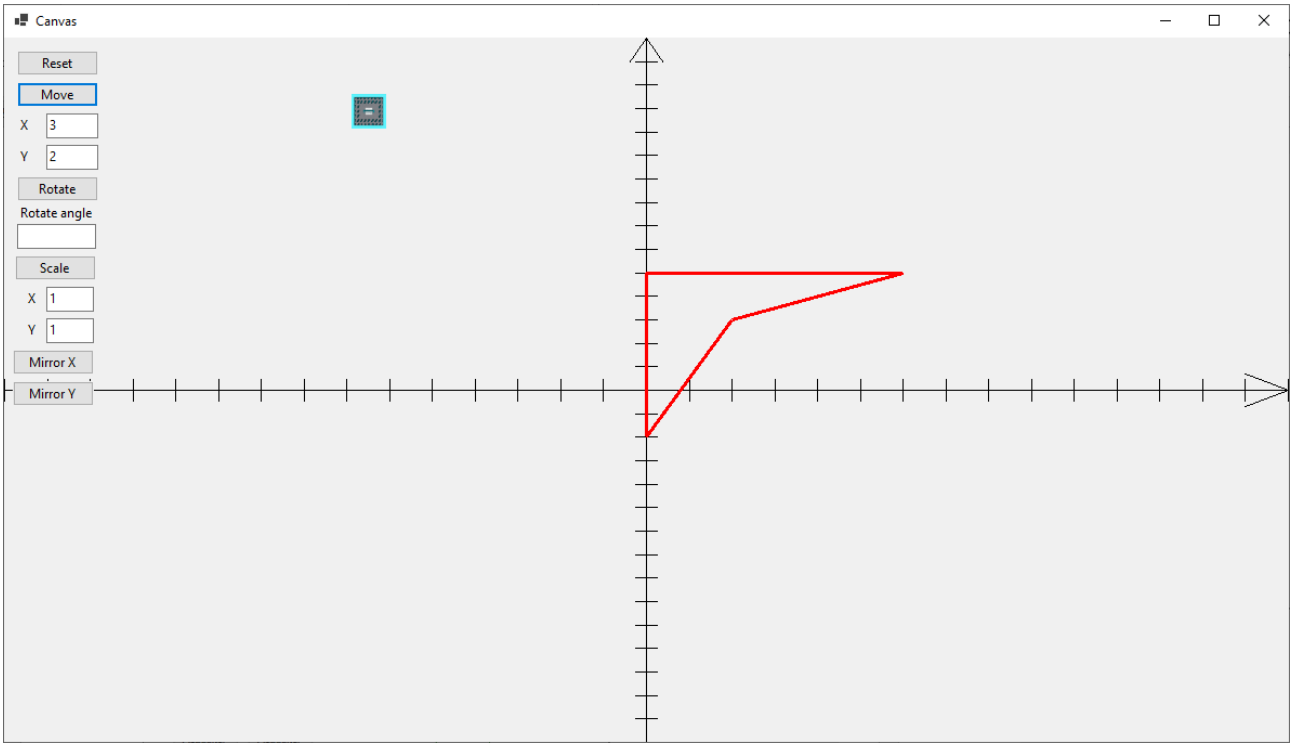


Рисунок 2 – Пернос

Рисунок 3 – Поворот

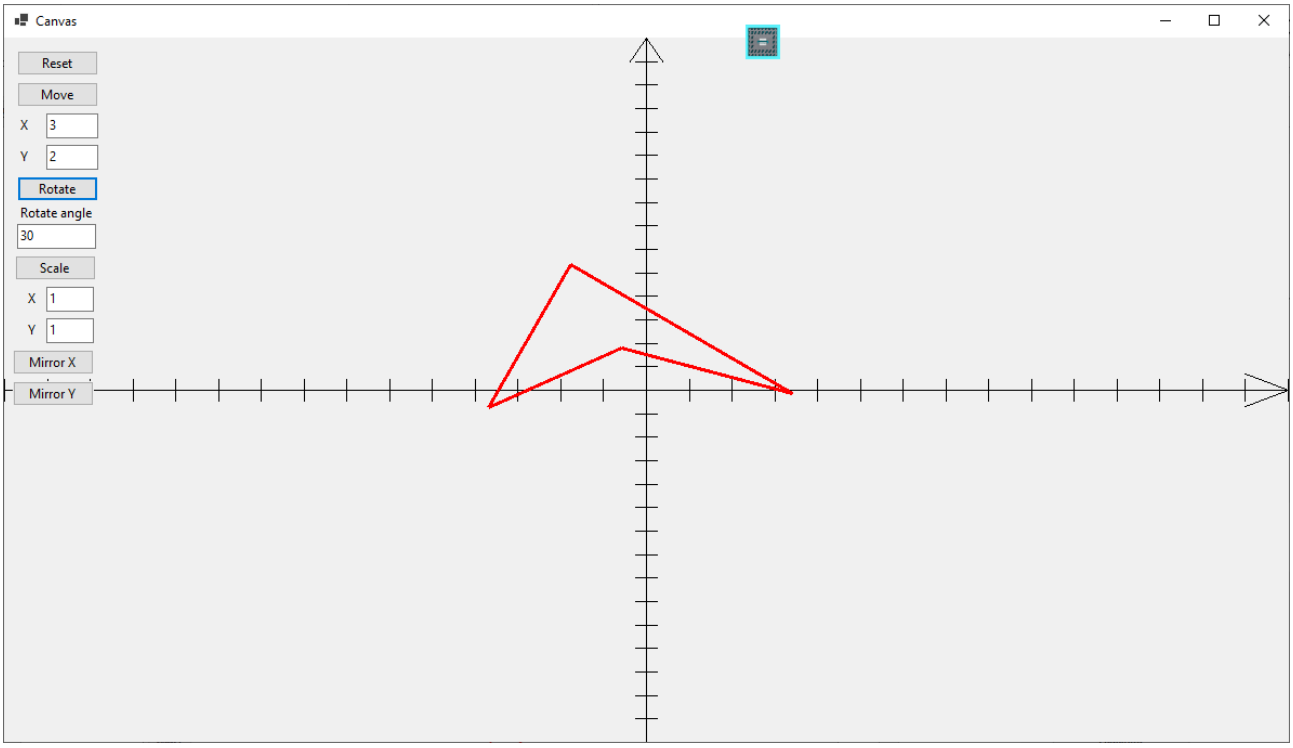


Рисунок 4 – Поворот

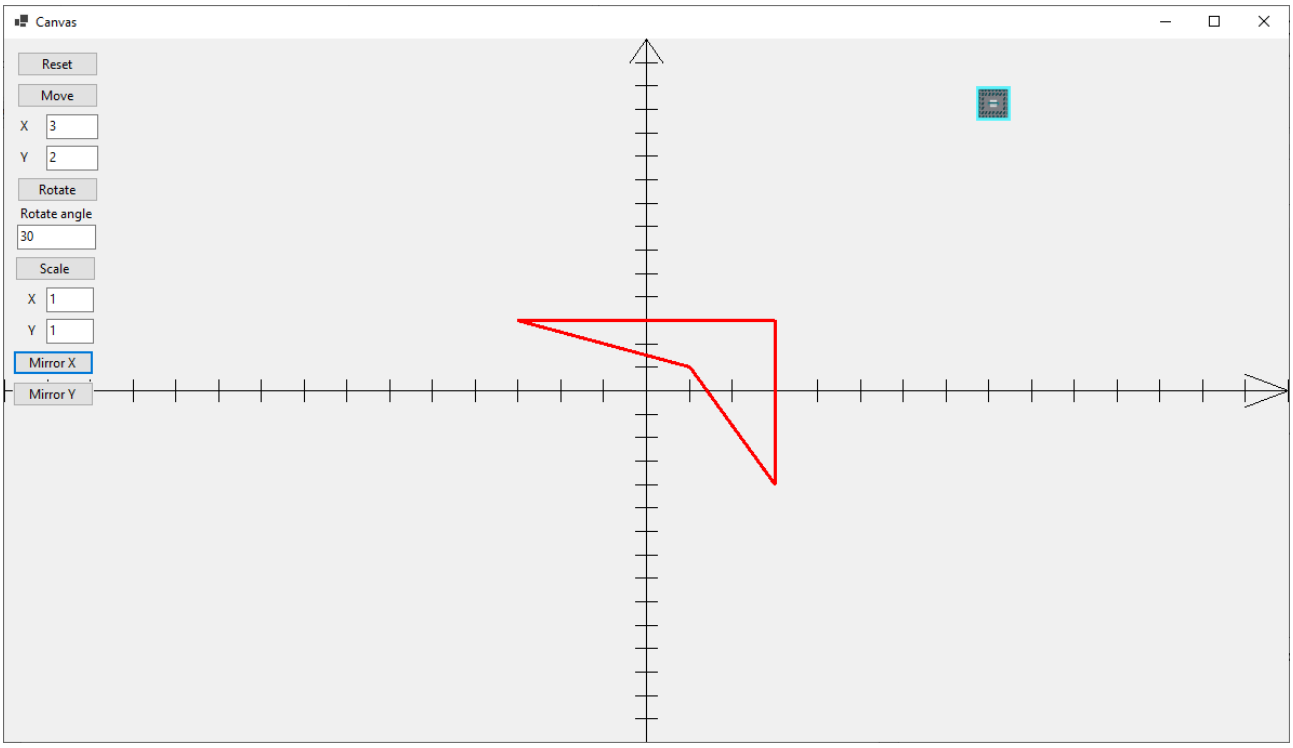


Рисунок 5 – Отражение по Ox

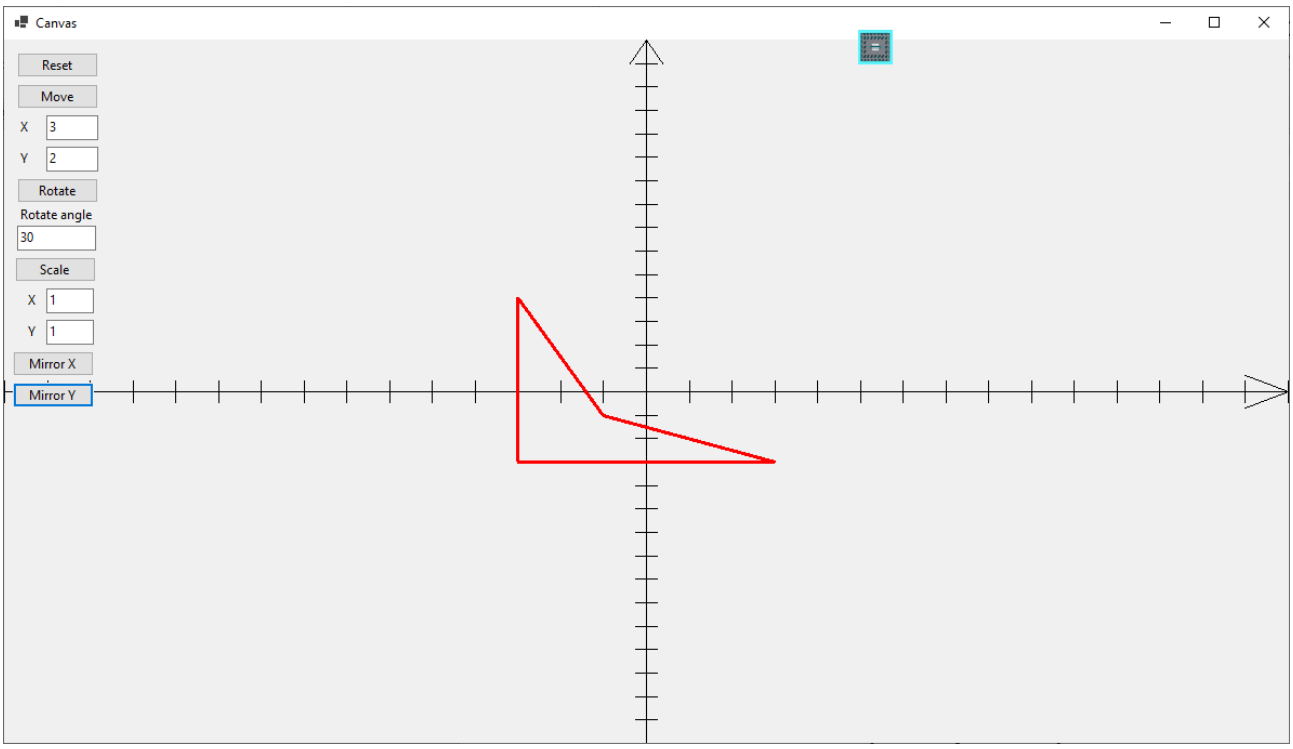


Рисунок 6 – Отражение по Oy