Министерство образования РБ

Учреждение образования

«Витебский государственный технологический университет»

Кафедра «Информационные системы и автоматизация производства»

Курсовая работа

по дисциплине «Системы компьютерной графики»

Разработка приложения для создания и преобразования графических объектов

Выполнил: студент группы ИТ-4

Маслов Евгений Сергеевич

Проверил: доцент

кафедры ИСАП

Надёжная Наталья Леонидовна

Витебск, 2017

Содержание

Изм.

Лист

№ докум.

Подпись

Дата

Лист

УО «ВГТУ» KР.015 1-40 05 01-01 ПЗ

Разраб.

Маслов Е.С.

Провер.

Надёжная Н.Л.

Реценз.

Н. Контр.

Утверд.

Содержание

Лит.

Листов

УО «ВГТУ» ИСАП гр. Ит-4

Введение

1 Анализ исходного графического объекта и методов преобразования фигур ……………………………………………………………………………….…с.4

1.1 Математическое описание операций преобразования плоских фигур……………………………………………………………………с.5

1.2 Выбор и обоснование языка программирования и среды разработки……………………………………………………………...с.9

1.3 Задание базовой фигуры…………………………………………..с.10

2 Разработка алгоритма работы программы…………………………………с.11

3 Описание основных компонентов программы и последовательности разработки……………………………………………………………………..с.14

4 Исходный код программы…………………………………………………с.15

5 Проверка корректности работы программы в различных режимах……..с.20

Заключение

Литература

**Введение**

Изм.

Лист

№ докум.

Подпись

Дата

Лист

УО «ВГТУ» KР.015 1-40 05 01-01 ПЗ

Разраб.

Маслов Е.С.

Провер.

Надёжная Н.Л.

Реценз.

Н. Контр.

Утверд.

Введение

Лит.

Листов

УО «ВГТУ» ИСАП гр. Ит-4

Данная курсовая работа выполнена на языке программирования высокого уровня C# с использованием компилятора Microsoft Visual Studio Community 2015.

Целью данной курсовой работы является создание приложения для преобразования графических объектов на языке программирования С#.

Исходя из цели курсовой работы, были поставлены следующие задачи:

1. Проанализировать исходный объект
2. Выбрать язык программирования
3. Разработать алгоритм работы программы
4. Написать код программы
5. Произвести тестирование программы

**1 Анализ исходного графического объекта и методов преобразования фигур**

Видовыми преобразованиями графических объектов называются преобразования, ведущие к изменению их местоположения, формы, размеров или ориентации. Видовые преобразования разделяются на базовые и комбинированные. Базовые преобразования отвечают только за какой-либо один вид преобразования, а комбинированные - осуществляют сразу несколько видов преобразований. К базовым видовым преобразованиям относятся перенос, масштабирование, отображение. К комбинированным преобразованиям относятся вращение и др.

Масштабирование - это преобразование, при котором объект изменяет свои размеры пропорционально заданным значениям.

Отображение - это такое видовое преобразование при котором объект отображается относительно оси Х или оси Y или относительно начала координат.

В практической реализации видовые преобразования представляют собой операции перемножения двух матриц.

**1.1 Математическое описание операций преобразования плоских фигур**

**Представление точек и общая схема преобразования с использованием матриц**

Точка на плоскости представляется двумя своими координатами Эту пару можно представить в виде матрицы размером называемой вектор-строка или матрицы  размером называемой вектор-столбец. Данные матрицы часто называют координатными векторами. Если геометрическое преобразование представить в виде матрицы, то результат преобразования точки можно представить следующей формулой[2]:

,

где М – матрица геометрического преобразования; Р – вектор-строка представляющая исходную точку; Р’- вектор-строка полученная в результате преобразования.

Другими словами мы представили применение геометрического преобразования как произведение матриц. Рассмотрим данное произведение подробнее.

т.е. координаты преобразованной точки формируются суммой исходных координат умноженных на соответствующие коэффициенты:

Если в матрице преобразования и то

Результат преобразования совпадает с исходными значениями.

**Преобразование масштабирования**

Положим , a**¹**0 и d **¹**0, тогда

Таким образом

Следовательно, мы получили преобразование масштабирования. Коэффициенты a и d являются масштабирующими коэффициентами по осям x и y. Обычно их обозначают Sx и Sy, а соответствующая матрица носит название матрицы масштабирования.

Если 0 < Sx = Sy < 1 имеет место сжатие, в противном случае при Sx = Sy > 1 расширение.

Для обращения преобразования необходимо произвести масштабирование с коэффициентами, обратными заданным. Обратная матрица представляется следующим образом:

При разрешении отрицательных значений коэффициентов в матрице масштабирования, наряду с масштабированием, происходит отображение объектов относительно различных осей. Положим Sx = -2, а Sy = 1.

Координата y останется неизменной, а координата x увеличится вдовое и поменяет свой знак, т.е. наряду с масштабированием произойдет отображение относительно оси y.

При единичных величинах коэффициентов масштабирования не будет, а знаки будут определять оси отображения. Матрица



определит отображение относительно оси х, а матрица



- отображение относительно оси y.

Если отклониться от условий масштабирования, положив

и тогда

Преобразование

приведет к перестановке координат , , что геометрически можно представить как отражение относительно прямой x=y, делящей первый квадрант на октанты. Преобразование



приводит к отображению относительно прямой .

**Преобразование поворота**

Рассмотрим следующее преобразование.



Его результат можно рассматривать как поворот на 90 градусов против часовой стрелки относительно начала координат. Нетрудно определить другие частные случаи поворота:

на 180 -и 270 градусов - .

Рассмотрим схему поворота относительно начала координат на произвольный угол. Введем следующие обозначения: r – радиус поворота. Отрезок ОР, соединяющий начала координат О с поворачиваемой точкой Р; **l** - угол между осью x и начальным положением отрезка ОР; **q** - угол, на который выполняется поворот. Определим начальное положение точки Р и ее положение после поворота Р’.

(1)

Применив формулы косинуса и синуса суммы, получим:

Раскрывая скобки и учитывая формулу (1) получаем:

т.е. точка после поворота имеет координаты

Переходя к матричному представлению, получаем матрицу преобразования поворота на произвольный угол **q** относительно начала координат:

Для обращения преобразования необходимо выполнить поворот в противоположную сторону на тот же угол.

**1.2 Выбор и обоснование языка программирования и среды разработки**

Для реализации курсового проекта был выбран C#. Для решения поставленной задачи оптимально использовать язык C#, который является языком высокого уровня и позволяет быстро и эффективно создавать приложения.

Для реализации данного графического объекта была выбрана среда программирования Microsoft Visual Studio Community 2015, поскольку она представляет широкие возможности для программирования приложений.

Достоинства C#:

1. C# создавался параллельно с каркасом Framework .Net и в полной мере учитывает все его возможности - как FCL, так и CLR;
2. C# является полностью объектно-ориентированным языком, где даже типы, встроенные в язык, представлены классами;
3. C# является мощным объектным языком с возможностями наследования и универсализации;
4. C# является наследником языков C/C++, сохраняя лучшие черты этих популярных языков программирования. Общий с этими языками синтаксис, знакомые операторы языка облегчают переход программистов от С++ к C#;
5. Сохранив основные черты своего великого родителя, язык стал проще и надежнее. Простота и надежность, главным образом, связаны с тем, что на C# хотя и допускаются, но не поощряются такие опасные свойства С++ как указатели, адресация, разыменование, адресная арифметика;
6. Благодаря каркасу Framework .Net, ставшему надстройкой над операционной системой, программисты C# получают те же преимущества работы с виртуальной машиной, что и программисты Java. Эффективность кода даже повышается, поскольку исполнительная среда CLR представляет собой компилятор промежуточного языка, в то время как виртуальная Java-машина является интерпретатором байт-кода;
7. Мощная библиотека каркаса поддерживает удобство построения различных типов приложений на C#, позволяя легко строить Web-службы, другие виды компонентов, достаточно просто сохранять и получать информацию из базы данных и других хранилищ данных;
8. Реализация, сочетающая построение надежного и эффективного кода, является немаловажным фактором, способствующим успеху C#.

**1.3 Задание базовой фигуры**

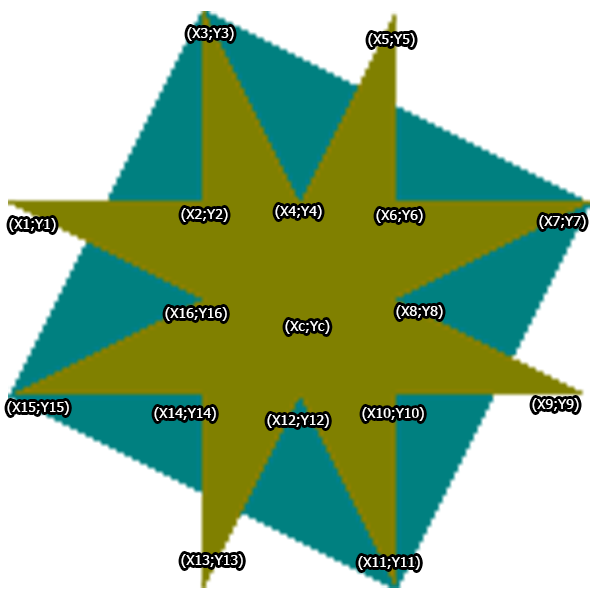


Рисунок 1.1 – Исходный графический объект

Исходный графический объект (рисунок 1.1) представляет собой два целых квадрата и четыре квадрата с вырезанными в них треугольниками. Объект обладает размером маленького квадрата величиной a.

Базовая фигура задается с помощью математических формул, которые описывают данные объекты, и методов задания объектов языка программирования С#.

Задний фон для фигуры реализован полигоном с заливкой в форме квадрата.

g.FillPolygon(br1, TrPoints);

Массив TrPoints – четыре точки квадрата, которые отсчитываются от центра фигуры по часовой стрелке. Вид границы сектора определяют значения свойства Реn, а вид заливки области, ограниченной линией границы - значения свойства Brush той поверхности , на которой метод рисует .

Для отрисовки передней части фигуры так же был использован полигон с заливкой. Фигуру разделили на прямоугольные треугольники и координаты их точек записали в массив.

g.FillPolygon(br2, curvePoints);

Где curvePoints – массив точек треугольников.

Для базовой фигуры точки выражаются через координаты центра и определяющей размер переменной mast. F означает не целое число в коде.

;

;

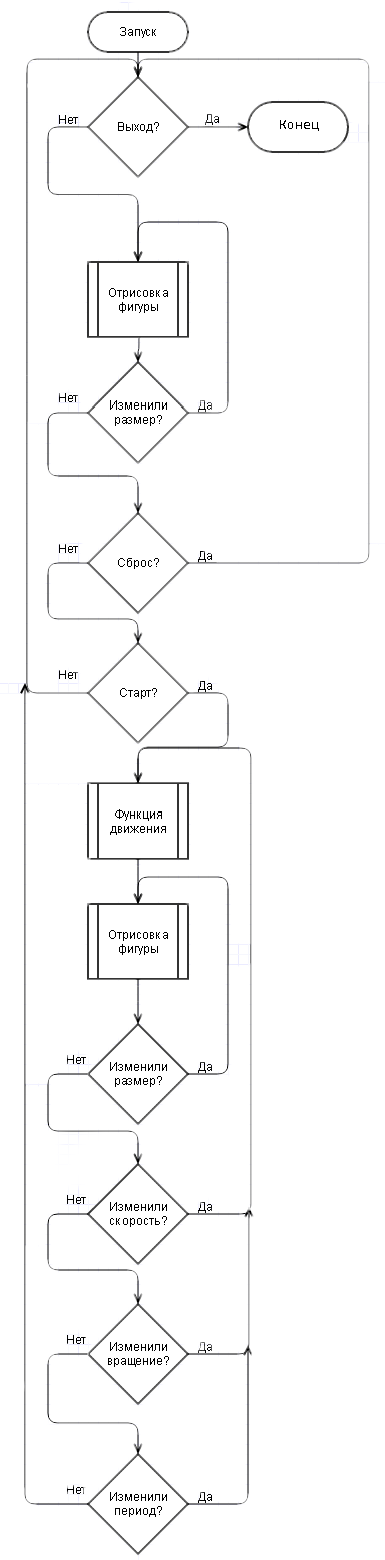
;

;

;

2. **РАЗРАБОТКА АЛГОРИТМА РАБОТЫ ПРОГРАММЫ**

Программа начинает свою работу. Производится задание значений, после чего с помощью класса FGRClass от событий кнопок в метод Ris передаются первоначальные значения данных. Из метода Ris в таймер передаются данные о местоположении объекта и его характеристика. В таймере происходит обработка данных и проверка корректности условий. В случае корректности условия таймер выполняет отрисовку объекта в соответствии с заданными значениями, в случае несоответствия условий блокирует действие. Процесс отрисовки цикличен, цикл приостанавливается при остановке таймера и возобновляется при его запуске. Закрытие программы выполняется корректно.



3. **ОПИСАНИЕ ОСНОВНЫХ КОМПОНЕНТОВ ПРОГРАММЫ И ПОСЛЕДОВАТЕЛЬНОСТИ РАЗРАБОТКИ**

Основные компоненты программы: PictureBox1, Timer1, Timer2, Button1, Button2, Button3, TrackBar1, TrackBar2, TrackBar3, TrackBar4, Label1, Label2, Label3, Label4

Timer1 - таймер запуска процесса движения

Timer2 - таймер для исполнения эмуляции вращения

Timer3 – таймер для остановки у границ

Button1 - кнопка запуска фигуры

Button2 - кнопка сброса настроек на настройки по умолчанию

Button3 - кнопка выхода из приложения

Label – отображение текста для пояснения

TrackBar1 – ползунок изменение размера

TrackBar2 – ползунок изменения скорости движения

TrackBar3 - ползунок изменения скорости вращения

TrackBar4 - ползунок изменение ширины периода

TrackBar5 – ползунок изменения времени задержки

В процессе разработки программы вначале производились вычисления для задания базовой фигуры, затем был выбран наиболее подходящий для практической реализации целей и задач язык программирования С#. Был разработан алгоритм работы программы, после чего произошло написание исходного кода.

4. **ИСХОДНЫЙ КОД ПРОГРАММЫ**

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 SKG\_KurRab

{

public partial class Form1 : Form {

//маштаб

int mast = 10;

//chastot

float T; float TT;

//направление движения

int dir = 1;

//приращение по х

float dx = 1; float dy = 1;

//скорость движения

int TransformSpeed;

//скорость вращения

float RotateSpeed;

//угол поворота

float Angle = 0;

//координаты центра изображения

float xc; double yc;

public Form1() { InitializeComponent(); }

private void Form1\_Load(object sender, EventArgs e) {

xc = Convert.ToInt32(pictureBox1.Width / 2);

yc = Convert.ToInt32(pictureBox1.Height / 2);

TransformSpeed = 20-trackBar2.Minimum;

RotateSpeed = trackBar3.Minimum;

T = trackBar4.Minimum;

TT = 0;

ris(); }

private int x1() {

return Convert.ToInt32( xc +((xc - 1.5f \* mast)-xc)\*Math.Cos(Angle)+(yc-(yc - 0.5f \* mast))\*Math.Sin(Angle));

}

private int y1() {

return Convert.ToInt32(yc + ((xc - 1.5f \* mast) - xc) \* Math.Sin(Angle) + ((yc - 0.5f \* mast) - yc) \* Math.Cos(Angle));

}

private int x2() {

return Convert.ToInt32(xc +((xc - 0.5f \* mast)-xc)\*Math.Cos(Angle)+(yc-(yc-0.5f\*mast))\*Math.Sin(Angle));

}

private int y2() {

return Convert.ToInt32(yc + ((xc - 0.5f \* mast) - xc) \* Math.Sin(Angle) + ((yc - 0.5f \* mast) - yc) \* Math.Cos(Angle));

}

private int x3() {

return Convert.ToInt32(xc +((xc - 0.5f \* mast)-xc)\*Math.Cos(Angle)+(yc-(yc - 1.5f \* mast))\*Math.Sin(Angle)); }

private int y3() {

return Convert.ToInt32(yc + ((xc - 0.5f \* mast) - xc) \* Math.Sin(Angle) + ((yc - 1.5f \* mast) - yc) \* Math.Cos(Angle)); }

private int x4() {

return Convert.ToInt32(xc +(xc-xc)\*Math.Cos(Angle)+(yc-(yc - 0.5f \* mast))\*Math.Sin(Angle)); }

private int y4() {

return Convert.ToInt32(yc + (xc - xc) \* Math.Sin(Angle) + ((yc - 0.5f \* mast) - yc) \* Math.Cos(Angle)); }

private int x5() {

return Convert.ToInt32(xc +((xc + 0.5f \* mast)-xc)\*Math.Cos(Angle)+(yc-(yc - 1.5f \* mast))\*Math.Sin(Angle)); }

private int y5() {

return Convert.ToInt32(yc + ((xc + 0.5f \* mast) - xc) \* Math.Sin(Angle) + ((yc - 1.5f \* mast) - yc) \* Math.Cos(Angle)); }

private int x6() {

return Convert.ToInt32(xc +((xc + 0.5f \* mast)-xc)\*Math.Cos(Angle)+(yc-(yc - 0.5f \* mast))\*Math.Sin(Angle)); }

private int y6() {

return Convert.ToInt32(yc + ((xc + 0.5f \* mast) - xc) \* Math.Sin(Angle) + ((yc - 0.5f \* mast) - yc) \* Math.Cos(Angle)); }

private int x7() {

return Convert.ToInt32(xc +((xc + 1.5f \* mast)-xc)\*Math.Cos(Angle)+(yc-(yc - 0.5f \* mast))\*Math.Sin(Angle)); }

private int y7() {

return Convert.ToInt32(yc + ((xc + 1.5f \* mast) - xc) \* Math.Sin(Angle) + ((yc - 0.5f \* mast) - yc) \* Math.Cos(Angle)); }

private int x8() {

return Convert.ToInt32(xc +((xc + 0.5f \* mast)-xc)\*Math.Cos(Angle)+(yc-yc)\*Math.Sin(Angle)); }

private int y8() {

return Convert.ToInt32(yc + ((xc + 0.5f \* mast) - xc) \* Math.Sin(Angle) + (yc - yc) \* Math.Cos(Angle)); }

private int x9() {

return Convert.ToInt32(xc +((xc + 1.5f \* mast)-xc)\*Math.Cos(Angle)+(yc-(yc + 0.5f \* mast))\*Math.Sin(Angle)); }

private int y9() {

return Convert.ToInt32(yc + ((xc + 1.5f \* mast) - xc) \* Math.Sin(Angle) + ((yc + 0.5f \* mast) - yc) \* Math.Cos(Angle)); }

private int x10() {

return Convert.ToInt32(xc +((xc + 0.5f \* mast)-xc)\*Math.Cos(Angle)+(yc-(yc + 0.5f \* mast))\*Math.Sin(Angle)); }

private int y10() {

return Convert.ToInt32(yc + ((xc + 0.5f \* mast) - xc) \* Math.Sin(Angle) + ((yc + 0.5f \* mast) - yc) \* Math.Cos(Angle)); }

private int x11() {

return Convert.ToInt32(xc +((xc + 0.5f \* mast)-xc)\*Math.Cos(Angle)+(yc-(yc + 1.5f \* mast))\*Math.Sin(Angle)); }

private int y11() {

return Convert.ToInt32(yc + ((xc + 0.5f \* mast) - xc) \* Math.Sin(Angle) + ((yc + 1.5f \* mast) - yc) \* Math.Cos(Angle));

}

private int x12() {

return Convert.ToInt32(xc + ((xc) - xc) \* Math.Cos(Angle) + (yc - (yc + 0.5f \* mast)) \* Math.Sin(Angle));

}

private int y12() {

return Convert.ToInt32(yc + ((xc) - xc) \* Math.Sin(Angle) + ((yc + 0.5f \* mast) - yc) \* Math.Cos(Angle));

}

private int x13() {

return Convert.ToInt32(xc +((xc - 0.5f \* mast)-xc)\*Math.Cos(Angle)+(yc-(yc + 1.5f \* mast))\*Math.Sin(Angle));

}

private int y13() {

return Convert.ToInt32(yc + ((xc - 0.5f \* mast) - xc) \* Math.Sin(Angle) + ((yc + 1.5f \* mast) - yc) \* Math.Cos(Angle));

}

private int x14() {

return Convert.ToInt32(xc +((xc - 0.5f \* mast)-xc)\*Math.Cos(Angle)+(yc-(yc + 0.5f \* mast))\*Math.Sin(Angle));

}

private int y14() {

return Convert.ToInt32(yc + ((xc - 0.5f \* mast) - xc) \* Math.Sin(Angle) + ((yc + 0.5f \* mast) - yc) \* Math.Cos(Angle));

}

private int x15() {

return Convert.ToInt32(xc +((xc - 1.5f \* mast)-xc)\*Math.Cos(Angle)+(yc-(yc + 0.5f \* mast))\*Math.Sin(Angle));

}

private int y15() {

return Convert.ToInt32(yc + ((xc - 1.5f \* mast) - xc) \* Math.Sin(Angle) + ((yc + 0.5f \* mast) - yc) \* Math.Cos(Angle));

}

private int x16() {

return Convert.ToInt32(xc +((xc - 0.5f \* mast)-xc)\*Math.Cos(Angle)+(yc-yc)\*Math.Sin(Angle));

}

private int y16() {

return Convert.ToInt32(yc + ((xc - 0.5f \* mast) - xc) \* Math.Sin(Angle) + (yc - yc) \* Math.Cos(Angle));

}

//кнопка старт

private void button1\_Click(object sender, EventArgs e) {

timer1.Start();

timer2.Start();

}

private void ris() {

//присваиваю g рисование графиков

Bitmap bmp = new Bitmap(pictureBox1.Width, pictureBox1.Height);

//Graphics g = pictureBox1.CreateGraphics();

Graphics g = Graphics.FromImage(bmp);

pictureBox1.Image = bmp;

//создаю ручку

Pen pen = new Pen(Color.Green);

//сброс

g.Clear(SystemColors.Control);

//точки фигуры для полигона

Point point1 = new Point(x1(), y1());

Point point2 = new Point(x2(), y2());

Point point3 = new Point(x3(), y3());

Point point4 = new Point(x4(), y4());

Point point5 = new Point(x5(), y5());

Point point6 = new Point(x6(), y6());

Point point7 = new Point(x7(), y7());

Point point8 = new Point(x8(), y8());

Point point9 = new Point(x9(), y9());

Point point10 = new Point(x10(), y10());

Point point11 = new Point(x11(), y11());

Point point12 = new Point(x12(), y12());

Point point13 = new Point(x13(), y13());

Point point14 = new Point(x14(), y14());

Point point15 = new Point(x15(), y15());

Point point16 = new Point(x16(), y16());

//массив из точек для полигона фигуры

Point[] curvePoints = {

point1, point2, point3, point4,

point5, point6, point7, point8,

point9, point10, point11, point12,

point13, point14, point15, point16, };

//массив фигуры для заднего фона

Point[] TrPoints = {

point3,

point7,

point11,

point15 };

//рисование заднего поле закрашенного

Brush br1 = new SolidBrush(Color.FromArgb(0, colorDec(), colorInc()));

Brush br2 = new SolidBrush(Color.FromArgb(colorInc(), colorDec(), 0));

g.FillPolygon(br1, TrPoints);

//рисование закрашенной фигуры

g.FillPolygon(br2, curvePoints); }

//размер фигуры

private void trackBar1\_Scroll(object sender, EventArgs e) {

mast = trackBar1.Value;

ris(); }

private void trackBar2\_Scroll(object sender, EventArgs e) {

TransformSpeed = 20 - Convert.ToInt32(trackBar2.Value);

ris(); }

private void trackBar3\_Scroll(object sender, EventArgs e) {

RotateSpeed = trackBar3.Value;

ris(); }

private void timer1\_Tick(object sender, EventArgs e) {

if (x1() > pictureBox1.Width || x2() > pictureBox1.Width ||

x3() > pictureBox1.Width || x4() > pictureBox1.Width ||

x5() > pictureBox1.Width || x6() > pictureBox1.Width ||

x7() > pictureBox1.Width || x8() > pictureBox1.Width ||

x9() > pictureBox1.Width || x10() > pictureBox1.Width ||

x11() > pictureBox1.Width || x12() > pictureBox1.Width ||

x13() > pictureBox1.Width || x14() > pictureBox1.Width ||

x15() > pictureBox1.Width || x16() > pictureBox1.Width) {

xc = xc - 2;

dx = -dx;

dir = -dir; }

if( y1() > pictureBox1.Height || y2() > pictureBox1.Height ||

y3() > pictureBox1.Height || y4() > pictureBox1.Height ||

y5() > pictureBox1.Height || y6() > pictureBox1.Height ||

y7() > pictureBox1.Height || y8() > pictureBox1.Height ||

y9() > pictureBox1.Height || y10() > pictureBox1.Height ||

y11() > pictureBox1.Height || y12() > pictureBox1.Height ||

y13() > pictureBox1.Height || y14() > pictureBox1.Height ||

y15() > pictureBox1.Height || y16() > pictureBox1.Height) {

yc = yc - 2;

dy = -dy; }

if ( x1() < 0 || x2() < 0 || x3() < 0 || x4() < 0 ||

x5() < 0 || x6() < 0 || x7() < 0 || x8() < 0 ||

x9() < 0 || x10() < 0 || x11() < 0 || x12() < 0 ||

x13() < 0 || x14() < 0 || x15() < 0 || x16() < 0) {

xc = xc + 2;

dx = -dx;

dir = -dir; }

if ( y1() < 0 || y2() < 0 || y3() < 0 || y4() < 0 ||

y5() < 0 || y6() < 0 || y7() < 0 || y8() < 0 ||

y9() < 0 || y10() < 0 || y11() < 0 || y12() < 0 ||

y13() < 0 || y14() < 0 || y15() < 0 || y16() < 0) {

yc = yc + 2;

dy = -dy; }

ris();

xc = xc + dx;

yc = (pictureBox1.Height / 2 + ((pictureBox1.Height - 3 \* mast) / 2) \* Math.Sin(21 \* (TT/10)))-2;

Angle += 0.01f \* RotateSpeed \* dir;

TT += 0.01f \* T; }

private void button2\_Click(object sender, EventArgs e) {

timer1.Stop();

timer2.Stop();

xc = Convert.ToInt32(pictureBox1.Width / 2);

yc = Convert.ToInt32(pictureBox1.Height / 2);

TransformSpeed = 20 - trackBar2.Minimum;

trackBar2.Value = trackBar2.Minimum;

RotateSpeed = trackBar3.Minimum;

trackBar3.Value = trackBar3.Minimum;

mast = trackBar1.Minimum;

trackBar1.Value = trackBar1.Minimum;

trackBar4.Value = trackBar4.Minimum;

T = trackBar4.Minimum;

ris(); }

private void button3\_Click(object sender, EventArgs e) {

Application.Exit(); }

private void timer2\_Tick(object sender, EventArgs e) {

timer1.Interval = TransformSpeed; }

private void trackBar4\_Scroll(object sender, EventArgs e) {

T = trackBar4.Value; }

private byte colorInc() {

try {

double col = 255 \* (xc - 1.5 \* mast) / (pictureBox1.Width - 1.5 \* mast - 1.5 \* mast);

return Convert.ToByte(col); }

catch { return 0; } }

private byte colorDec() {

try {

double col = -255 \* (xc - 1.5 \* mast) / (pictureBox1.Width - 1.5 \* mast - 1.5 \* mast) + 255;

return Convert.ToByte(col); }

catch { return 0; } } }}

1. **ПРОВЕРКА КОРРЕКТНОСТИ РАБОТЫ ПРОГРАММЫ В РАЗЛИЧНЫХ РЕЖИМАХ**
2. Рабочее окно программы с прорисованным объектом изображено на рисунке 5.1

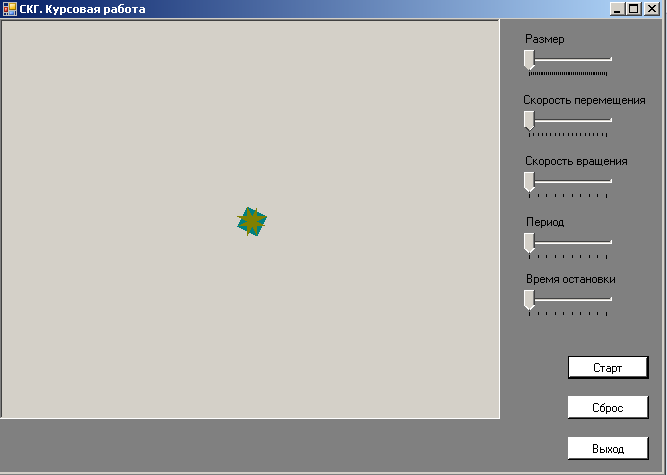


Рисунок 5.1 – Рабочее окно программы

На рисунке 5.2 изображено масштабирование объекта

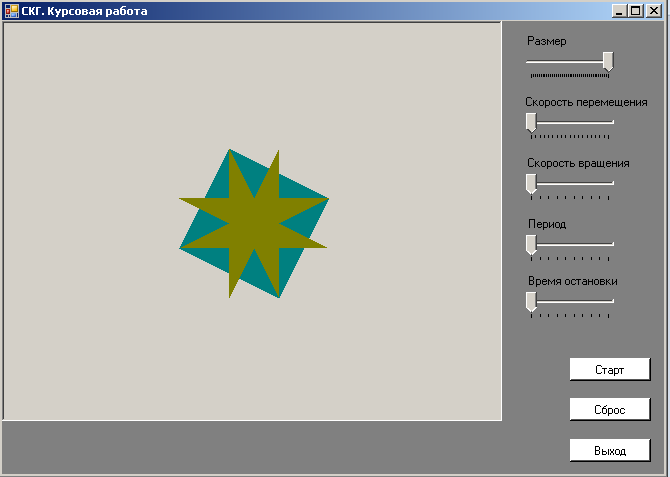


Рисунок 5.2 ­– Рабочее окно программы с измененным размером фигуры

3. На рисунке 5.3 показан процесс вращения фигуры

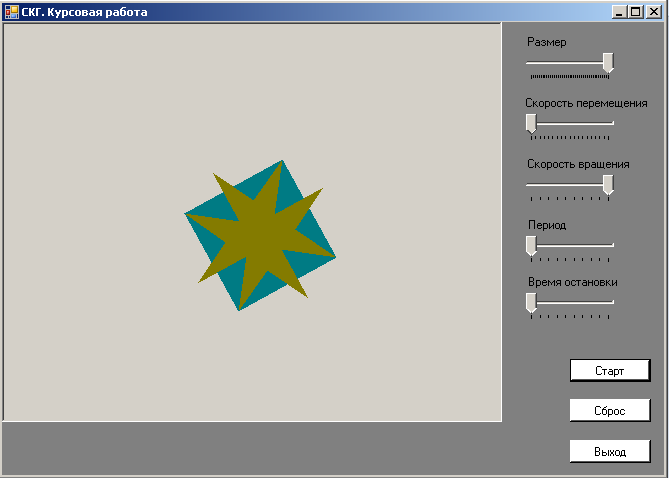


Рисунок 5.3 – Рабочее окно программы с измененным размером и

скоростью вращения фигуры

4. На рисунке 5.4 показан процесс перемещения фигуры, изменение его цвета и частоты периода

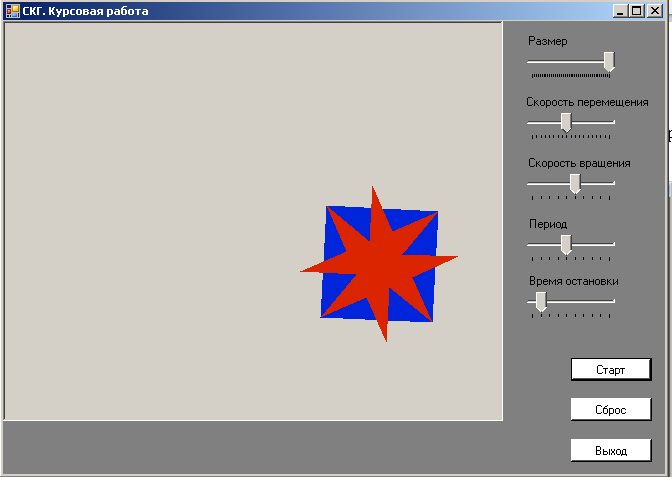


Рисунок 5.4 – ­­Рабочее окно программы, показывающее изменение

периода и цвета фигуры от координат

**Заключение**

В результате данного курсового проекта было разработано графическое приложение, которое непосредственно показывает возможности создания и преобразования графических объектов с помощью языка программирования язык C#, который является языком высокого уровня и позволяет быстро и эффективно создавать приложения.

Для реализации проекта был разработан специальный алгоритм работы, следуя которому в итоге были решены все задачи, поставленные изначально.

Также, при тестировании программы, неточности не обнаружены, программа работает корректно в разных режимах.

**Список использованной литературы**

1. Герберт, Ш.Г. C#3.0 Руководство для начинающих / Герберт, Ш.Г. – издательский дом Вильямс, 2009. – 688 с.
2. Постнов, К.В. Компьютерная графика: учеб. курс: М.: МГСУ. 2009. - 249 с
3. Боресков, А.В. Компьютерная графика: Учебник и практикум для прикладного бакалавриата / А.В. Боресков, Е.В. Шикин. - Люберцы: Юрайт, 2016. - 219 c.
4. Visual C# 2008 Базовый курс/К. Уотсон [и др.]; под общ. ред. К. Нейгеля. – издательский дом Вильямс, 2009. – 1216 с.
5. С# 2008 и платформа .NET 3.5 для профессионалов/ К. Нейгел [и др.]; под общ. ред. Кристиан Нейгеля. – издательский дом Вильямс, 2008. – 1392 с.