Министерство образования РБ

Учреждение образования «Витебский государственный технологический университет»

Курсовая работа

по дисциплине «Системы компьютерной графики»

Разработка приложения для создания и преобразования графических объектов

Выполнила: студентка группы ИТ-9

Радайкина Екатерина Дмитриевна

Проверил: доцент

кафедры ИСАП

Дунина Елена Брониславовна

Витебск

2020

Содержание

Изм.

Лист

№ докум.

Подпись

Дата

Лист

УО «ВГТУ» KР.10 1-40 05 01-01 ПЗ

Разраб.

Радайкина Е,Д.

Провер.

Дунина Е.Б.

Реценз.

Н. Контр.

Утверд.

Содержание

Лит.

Листов

УО «ВГТУ» ИСАП гр. Ит-9

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.11

2.Разработка алгоритма работы программы……………………………….с.13

3. Описание основных компонентов программы и последовательности разработки………………………………………………………………….......с.15

4.Исходный код программы…………………………………………………с.16

5. Проверка корректности работы программы в различных режимах……с.21

Заключение

Литература

Введение

Изм.

Лист

№ докум.

Подпись

Дата

Лист

УО «ВГТУ» KР.10 1-40 05 01-01 ПЗ

Разраб.

Радайкина Е.Д.

Провер.

Дунина Е.Б.

Реценз.

Н. Контр.

Утверд.

Введение

Лит.

Листов

УО «ВГТУ» ИСАП гр. Ит-9

Данный проект подразумевает собой создание графического приложения, которое эмулирует движение базовой фигуры таким образом, чтобы ее центр описывал прямолинейную траекторию. При движении вдоль траектории фигура одновременно вращается вокруг центральной точки, а также меняет цвет. Скорость движения фигуры вдоль траектории, а также скорость вращения вокруг центральной точки должна задаваться пользователем. Необходимо предусмотреть возможность масштабирования фигуры без изменения пропорций и возможностью возврата к исходному размеру.

Запуск и остановка режима эмуляции движения, а также все параметры данного режима задаются пользователем при помощи графических элементов интерфейса, а именно, панели инструментов.

Для реализации проекта мной будет использоваться язык программирования высокого уровня **C#,** с использованием среды разработки Visual Studio 2019.

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**

В данном курсовом проекте была поставлена задача: Написать Windows-приложение, которое эмулирует движение базовой фигуры.

Движение фигуры и изменение её скорости обрабатывается таймером.

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 переноса и однородные координаты

В ходе обработки модели часто возникает необходимость переместить объект, не изменяя его размеров и формы. Его направление определяет направление переноса, а проекции на координатные оси задают перемещение вдоль этих осей:

,где и - проекции вектора на оси x иy

**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:



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’[2].

(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[2]:



**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[2].



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[2]:



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

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 x’= y, y’= x, что геометрически можно представить, как отражение относительно прямой x=y, делящей первый квадрант на октанты. Преобразование



приводит к отображению относительно прямой y = -x.

**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.

C# — элегантный, типо безопасный объектно-ориентированный язык, предназначенный для разработки разнообразных безопасных и мощных приложений, выполняемых в среде .NET Framework. С помощью языка C# можно создавать обычные приложения Windows, XML-веб-службы, распределенные компоненты, приложения "клиент-сервер", приложения баз данных и т. д. Visual C# предоставляет развитый редактор кода, конструкторы с удобным пользовательским интерфейсом, встроенный отладчик и множество других средств, упрощающих разработку приложений на базе языка C# и .NET Framework.

Как объектно-ориентированный язык, C# поддерживает понятия инкапсуляции, наследования и полиморфизма. Все переменные и методы, включая метод Main – точку входа приложения – инкапсулируются в определения классов. Класс может наследовать непосредственно из одного родительного класса, но может реализовывать любое число интерфейсов.

C# создавался как язык компонентного программирования, и в этом одно из главных достоинств языка, направленное на возможность повторного использования созданных компонентов. Из других объективных факторов отметим следующие:

Для реализации данного графического объекта была выбрана среда программирования Microsoft Visual Studio Community 2019, поскольку она представляет широкие возможности для программирования приложений.

Microsoft Visual Studio 2019 — это набор инструментов для создания программного обеспечения: от планирования до разработки пользовательского интерфейса, написания кода, тестирования, отладки, анализа качества кода и производительности, развертывания в средах клиентов и сбора данных телеметрии по использованию. Эти инструменты предназначены для максимально эффективной совместной работы; все они доступны в интегрированной среде разработки (IDE) Visual Studio.

Visual Studio можно использовать для создания различных типов приложений, от простых приложений для магазина и игр для мобильных клиентов до больших и сложных систем, обслуживающих предприятия и центры обработки данных. Вы можете создавать:

1. приложения и игры, которые выполняются не только на платформе Windows, но и на Android и iOS;
2. веб-сайты и веб-службы на основе ASP.NET, JQuery, AngularJS и других популярных платформ;
3. приложения для самых разных платформ и устройств, включая, но не ограничиваясь: Office, Sharepoint, Hololens, Kinect и "Интернета вещей";
4. игры и графические приложения для разных устройств Windows, включая Xbox, с поддержкой DirectX.

**1.3 Задание базовой фигуры**

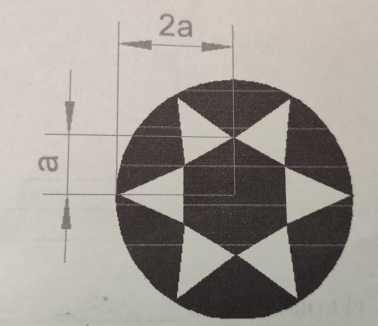


Рисунок 1.1 - Исходный графический объект.

Базовая фигура представляет собой шестиконечную звезду, которая вписана в окружность.

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

Задний фон для фигуры реализован заливкой в форме круга.

graph.FillEllipse(brSolid, x0, y0, diametr, diametr)

brSolid – цвет заливки.

x0, y0 – координаты центра окружности.

Для отрисовки передней части фигуры был использован полигон с заливкой. Фигуру разделили на треугольники.

graph.FillPolygon(brSoft, treeDots);

Где treeDots – массив точек треугольников.

Для базовой фигуры точки выражаются через координаты центра и opisR.

Формируем первый треугольник:

Изм.

Лист

Подпись

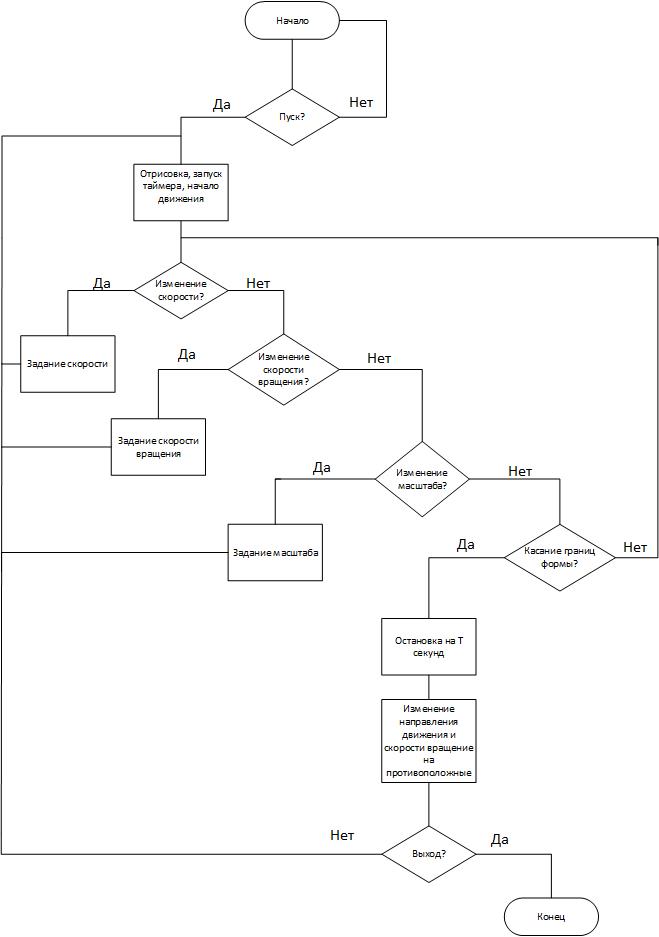
Дата

Лист

Далее вращаем его по окружности относительно ее центра на 60 градусов 4 раза.

**2**. **РАЗРАБОТКА АЛГОРИТМА РАБОТЫ ПРОГРАММЫ**

1. Прорисовка базовой фигуры путем вычисления координат всех контрольных точек, затем последовательного соединения их прямыми линиями, а также заливка областей.
2. Реализация движения фигуры по параболе из некоторой начальной точки.
3. Реализация последовательного изменения цвета заштрихованной и не заштрихованной областей базовой фигуры.
4. Реализация вращения базовой фигуры относительно ее центра.
5. Реализация возможности остановки движения в конечной и начальной точках движения ,на определенное время ,которое указывает пользователь.
6. Реализация возможности изменения масштаба, скорости движения и вращения базовой фигуры.



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

Основные компоненты программы: PictureBox, Timer, Button1(Draw), CheckBox1(StopSreed), TextBox1(X), TextBox2(Y) TrackBar1(Size), TrackBar2(SpeedFig), TrackBar3(SpeedRotete),Label(1-6):

Timer1 - Таймер запуска процесса движения и вращения

Button1(Draw)–Кнопка ввода координат фигуры

Button2(Restart)– Кнопка сброса параметров

CheckBox1(StopSreed) – Кнопка для остановки фигуры

TextBox1(X) – Окно для ввода координаты *Х*

TextBox1(Y)– Окно для ввода координаты *Y*

Label – Отображение текста для пояснения

TrackBar1(Size) – Ползунок изменение размера

TrackBar2(SpeedFig) – Ползунок изменения скорости движения

TrackBar3(SpeedRotete)– Ползунок изменения скорости вращения

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;

using System.Threading;

namespace Курсовой\_Проект

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

Figura();

textBox1.Text = Convert.ToString(pictureBox1.Width);

textBox2.Text = Convert.ToString(pictureBox1.Height);

Width = pictureBox1.Width;

Height = pictureBox1.Height;

}

int Width;

int Height;

double dt = 0; //у границ рабочей области

private float diametr = 40;

private void Figura()

{

float x0 = 0 + dx;

float y0 = pictureBox1.Height - diametr - dy;

{

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

Graphics graph = Graphics.FromImage(Ramka);

byte blz = (byte)(255 - 195 / (pictureBox1.Width - diametr) \* dx);//изменение цвета относительно x

byte rdz = (byte)(60 + 195 / (pictureBox1.Width - diametr) \* dx);

Color col = Color.FromArgb(255, blz, 60,rdz);

Color col1 = Color.FromArgb(255, rdz, 60, blz);

Pen pen;

if (diametr > 60) { pen = new Pen(col, 2); } else { pen = new Pen(col, 1); }

SolidBrush brSolid = new SolidBrush(col);

SolidBrush brSoft = new SolidBrush(col1);

graph.DrawRectangle(pen, 1, 1, pictureBox1.Width - 2, pictureBox1.Height - 2);//ramka

graph.FillEllipse(brSolid, x0, y0, diametr, diametr);//large circle

//построение звезды

float xz0 = x0 + diametr / 2;//центр фигруы

float yz0 = y0 + diametr / 2;

float opisR = (float)((diametr / 2) / (Math.Sqrt(5) - 1));//вписанный радиус (для нижних точек)

double starSide = (diametr / 2 \* Math.Sqrt((5 - Math.Sqrt(5)) /

2));//длина стороны , для нижних точек

PointF p1 = new PointF(xz0, yz0 - diametr / 2);//первая часть звезды

PointF p2 = new PointF((float)(xz0 + starSide / 4), yz0 - opisR / 2);

PointF p3 = new PointF((float)(xz0 - starSide / 4), yz0 - opisR / 2);

p1 = RotatePoint(p1, 90 \* yg, xz0, yz0);//смещение треугольника

p2 = RotatePoint(p2, 90 \* yg, xz0, yz0);

p3 = RotatePoint(p3, 90 \* yg, xz0, yz0);

for (int n = 0; n < 6; n++)

{//остальные части относительно первой

PointF pNew1 = RotatePoint(p1, 60 \* n, xz0, yz0);

PointF pNew2 = RotatePoint(p2, 60 \* n, xz0, yz0);

PointF pNew3 = RotatePoint(p3, 60 \* n, xz0, yz0);

PointF[] treeDots = { pNew1, pNew2, pNew3 };

graph.FillPolygon(brSoft, treeDots);

}

pictureBox1.Image = Ramka;

}

}

private PointF RotatePoint(PointF point, double n, float x0, float y0)

{//функция для смещения относительно точки на угол

float ddX = point.X - x0;

float ddY = point.Y - y0;

point.X = (float)(ddX \* Math.Cos(Math.PI \* n / 180) - ddY \* Math.Sin(Math.PI \* n / 180)) + x0;

point.Y = (float)(ddX \* Math.Sin(Math.PI \* n / 180) + ddY \* Math.Cos(Math.PI \* n / 180)) + y0;

return point;

}

float dx = 1;//смещение по координатам

float dy = 1;

float smesh = 3.1f;//скорость смещение по координтам

float proverka = 1;//чтобы проверка на выход за предел работала

double yg = 0;//угловое смещение

double ygSpeed = 0.01f;

private double FomA;

private double FomB;

private double FomC;

private void CalcFom()

{

double x1 = 0; double x2 = (pictureBox1.Width) / 2 - diametr / 2;

double y1 = (pictureBox1.Height - diametr); double y2 = 2;

FomA = Math.Sqrt((y1 - y2) / Math.Pow(x1 / x2 - 1, 2));

FomB = pictureBox1.Height - diametr - y2;

FomC = FomA / x2;

}

private float move(float x)

{

CalcFom();

return (float)(-Math.Pow(FomC \* x - FomA, 2) + FomB);

}

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

{

if (timer1.Enabled)

{

timer1.Stop();

button1.Text = "Старт";

}

else

{

button1.Text = "Стоп";

timer1.Start();

}

if (Convert.ToInt32(textBox1.Text) < Width && Convert.ToInt32(textBox1.Text) > 1)

{

pictureBox1.Width = Convert.ToInt32(textBox1.Text);

}

else { pictureBox1.Width = Width; }

if (Convert.ToInt32(textBox2.Text) < Height && Convert.ToInt32(textBox1.Text) > 1)

{

pictureBox1.Height = Convert.ToInt32(textBox2.Text);

}

else { pictureBox1.Height = Height; }

textBox1.Visible = false;

textBox2.Visible = false;

}

private void label3\_Click(object sender, EventArgs e)

{

PicDef();

Figura();

}

private void PicDef()

{

dx = dy = 0;

yg = 0;

timer1.Stop();

button1.Text = "Start";

label2.Text = "x";

label3.Text = "y";

label3.Text = "move";

textBox1.Visible = true;

textBox2.Visible = true;

}

private void timer1\_Tick\_1(object sender, EventArgs e)

{

Figura();

if (dx+trackBar2.Value >= pictureBox1.Width)

{

Thread.Sleep(1000\*Convert.ToInt32(textBox3.Text));

}

else

{

if (dx < 0)

Thread.Sleep(1000\*Convert.ToInt32(textBox3.Text));

}

if (proverka < 0)

{

dx -= smesh;

dy = move(dx);

proverka = -dy;

yg -= ygSpeed;

}

else

{

dx += smesh;

dy = move(dx);

proverka = dy;

yg += ygSpeed;

}

label2.Text = "x: " + Convert.ToString(dx);

label3.Text = "y: " + Convert.ToString(dy);

}

int t = 1;

private int prist() { t = Convert.ToInt32(textBox1.Text) - 2; return t; }

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

{

if (t <= 0) { timer1.Start(); timer2.Stop(); }

t -= 1;

}

private void trackBar1\_Scroll\_1(object sender, EventArgs e)

{

smesh = (trackBar1.Value) / 10;

}

private void trackBar2\_Scroll\_1(object sender, EventArgs e)

{

diametr = trackBar2.Value;

if (!timer1.Enabled)

{

Figura();//рисует фигуру

}

}

private void trackBar3\_Scroll\_1(object sender, EventArgs e)

{

ygSpeed = (double)(trackBar3.Value) / 100;

}

}

}

1. **ПРОВЕРКА КОРРЕКТНОСТИ РАБОТЫ ПРОГРАММЫ В РАЗЛИЧНЫХ РЕЖИМАХ**

3

1.На рисунке 5.1. изображён процесс отрисовки фигуры по заданным пользователем координатам работает исправно.

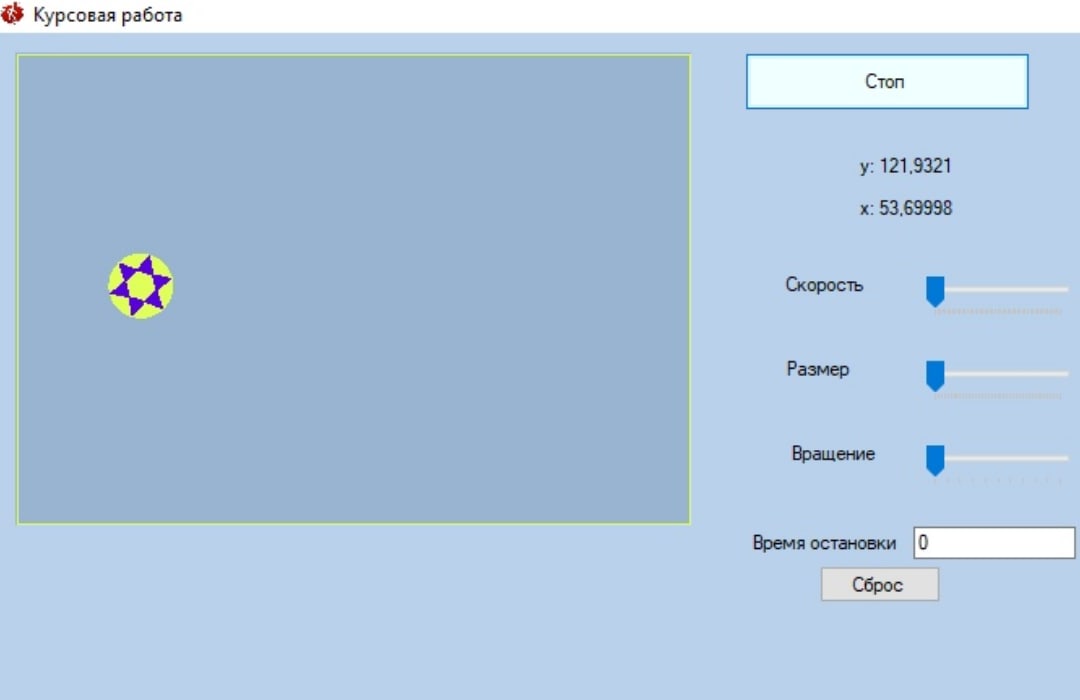


Рисунок 5.1 – Рабочее окно программы с заданным положение фигуры.

2. На рисунке 5.2. изображено масштабирование фигуры.

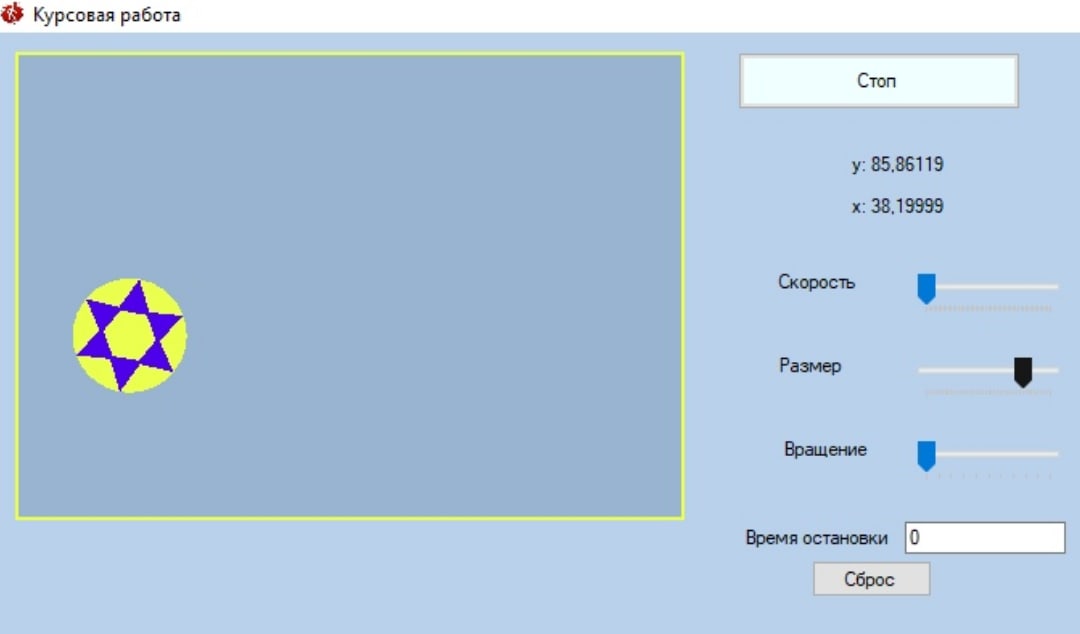


Рисунок 5.2 –Рабочее окно программы с увеличенным масштабом фигуры.

3. На рисунке 5.3. изображено увеличение скорости движения фигуры.

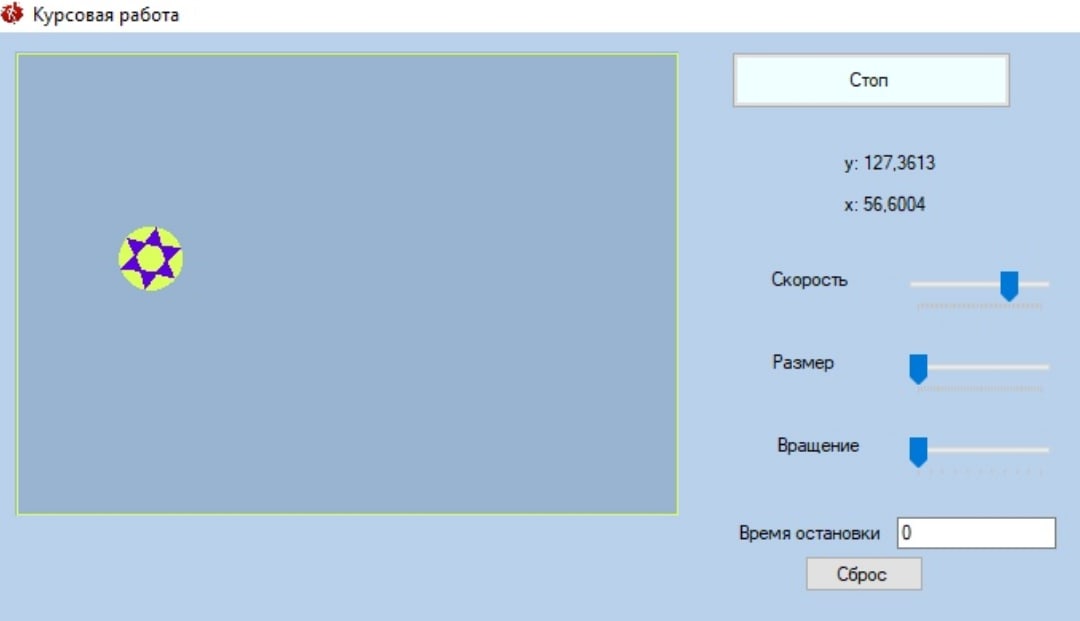


Рисунок 5.3 –Рабочее окно программы с увеличенной скоростью движения фигуры.

2. На рисунке 5.4. изображено увеличение скорости вращения фигуры.

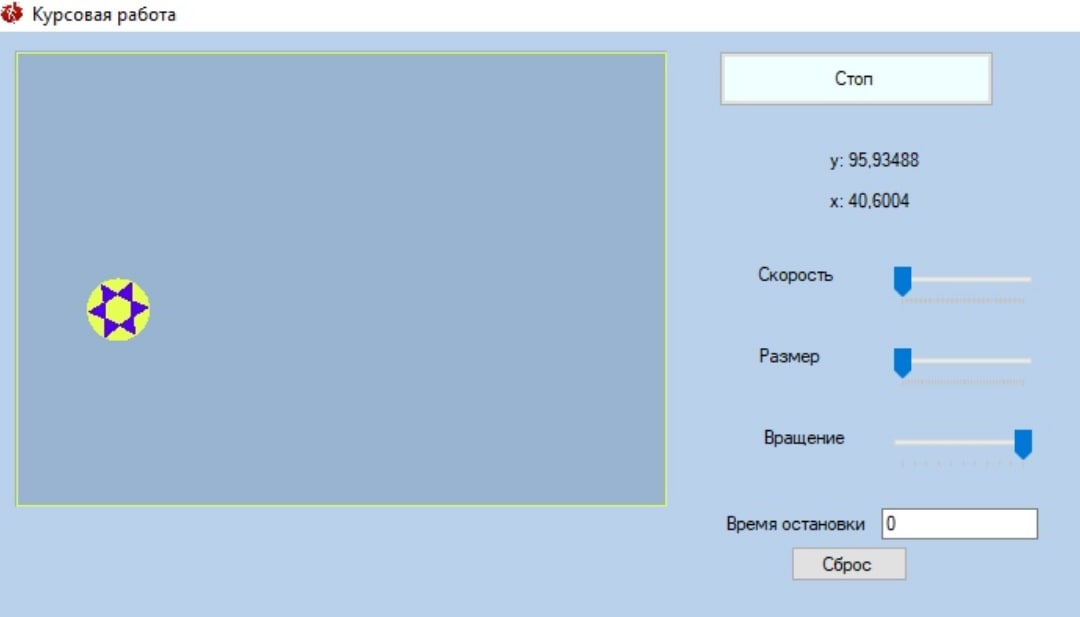


Рисунок 5.4 –Рабочее окно программы с увеличенной скоростью вращения фигуры.

2. На рисунке 5.5. изображено изменение цвета фигуры.

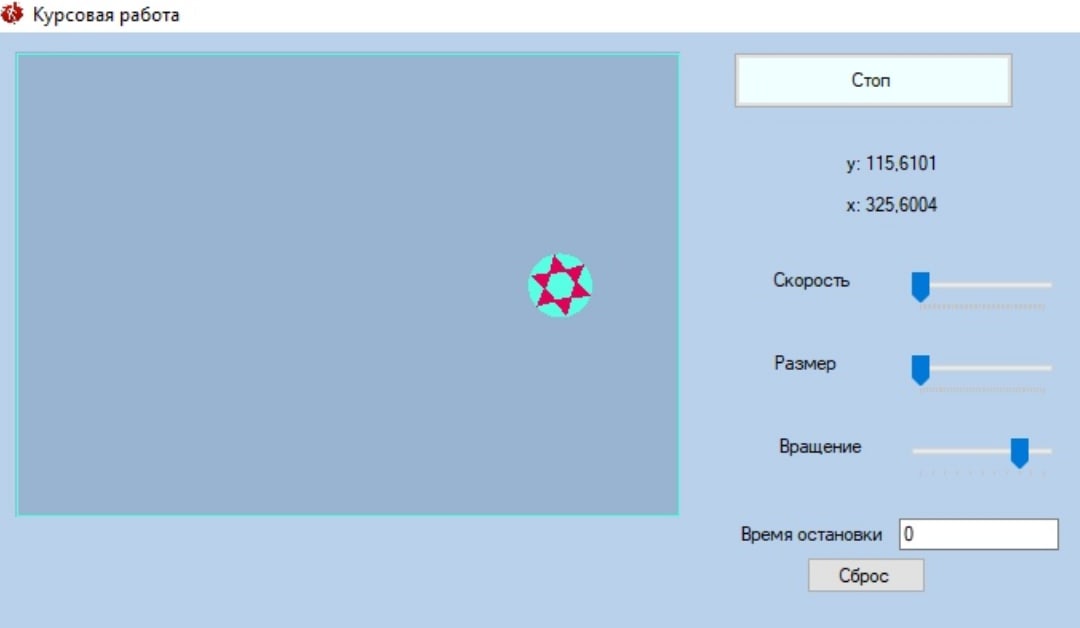


Рисунок 5.5 –Рабочее окно программы с изменённым цветом фигуры.

2. На рисунке 5.6. изображена остановка фигуры.

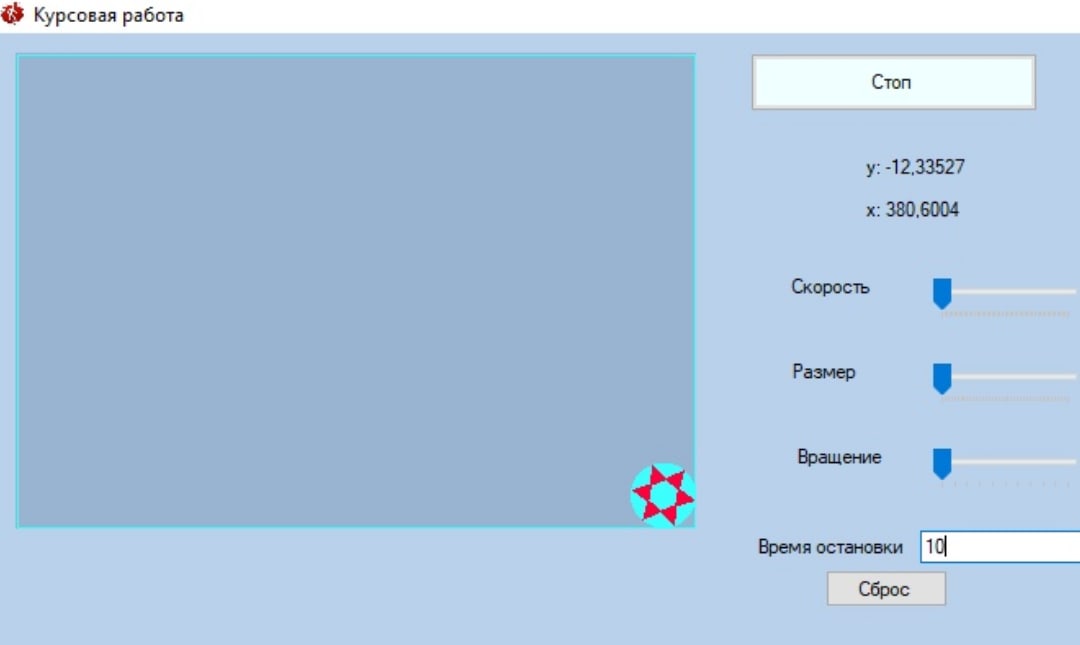


Рисунок 5.6 – Рабочее окно программы с остановленной фигурой.

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. Боресков, А.В. Компьютерная графика: Учебник и практикум для прикладного бакалавриата / А.В. Боресков, Е.В. Шикин. - Люберцы: Юрайт, 2016. - 219 c.
  2. Постнов,К.В. Компьютерная графика: учеб. курс: М.: МГСУ. 2009 - 249 с

3. С# 2008 и платформа .NET 3.5 для профессионалов (КристианНейгел и др.)

4. Visual C# 2008 Базовый курс (К. Уотсон, К. Нейгел и др).

5. C#3.0 Руководство для начинающих (Герберт Шилдт)

6.C# 2005 и платформа .NET 3.0 + CD с примерами и главами не вошедшими к печатному изданию (КристианНейгл, Билл Ивьен, ДжейлГлинн, Морган Скиннер)