Министерство образования Республики Беларусь

Учреждение образования

«Витебский государственный технологический университет»

Кафедра «Информационные системы и автоматизация производства»

КУРСОВАЯ РАБОТА

по дисциплине «Системы компьютерной графики»

на тему «Разработка приложения для создания и преобразования графических объектов»

Выполнил:

студент ФИТР

группы ИТ-8 \_\_\_\_\_\_\_\_\_\_ Борисенко А.В.

*подпись* *Ф.И.О.*

Проверил:

Руководитель

*должность*

доцент, кандидат физико-математических наук Дунина Е. Б.

*ученая степень, ученое звание* *Ф.И.О.*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*отметка о допуске к защите*

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_20\_\_\_г.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*дата* *подпись руководителя*

Витебск 2020

Содержание

Изм.

Лист

№ докум.

Подпись

Дата

Лист

3

УО «ВГТУ» KР.29 1-40 05 01-01 ПЗ

Разраб.

Борисенко А.В.

.

Провер.

Дунина Е. Б.

Реценз.

Н. Контр.

Утверд.

Содержание

Лит.

Листов

27

УО «ВГТУ» ИСАП гр. Ит-8

Введение

1 Анализ исходного графического объекта и методов преобразования фигур ……………………………………………………………………………….…...5

1.1 Математическое описание операций преобразования плоских фигур……………………………………………………………………...6

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

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

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

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

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

5 Проверка корректности работы программы в различных режимах……….24

Заключение

Литература

**Введение**

Изм.

Лист

№ докум.

Подпись

Дата

Лист

4

УО «ВГТУ» KР.29 1-40 05 01-01 ПЗ

Разраб.

Борисенко А.В

Провер.

Дунина Е. Б.

Реценз.

Н. Контр.

Утверд.

Введение

Лит.

Листов

27

УО «ВГТУ» ИСАП гр. Ит-8

Цель курсовой работы заключается в создании Windows-приложение, эмулирующего движение базовой фигуры таким образом, чтобы ее центр описывал заданную траектория. При движении вдоль траектории фигура одновременно вращается вокруг центральной точки, а также меняет цвет незаштрихованной части фигуры от синего (в начале траектории) до красного (в конце траектории), заштрихованных элементов – от желтого до синего.

Также в данной работе необходимо предусмотреть возможность изменения пользователем скорости вращения и движения фигуры и возможность масштабирования фигуры с сохранением ее пропорций и возможностью возвращения к первоначальному размеру, и начальное направление траектории в виде вектора перемещения. Данные возможности будут реализованы с помощью графических элементов интерфейса.

Для реализации проекта будет использоваться язык программирования C#, с применением 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**

Видовыми преобразованиями графических объектов называются преобразования, ведущие к изменению их местоположения, формы, размеров. Видовые преобразования разделяются на базовые и комбинированные. Базовые преобразования отвечают только за какой-либо один вид преобразования (перенос, масштабирование), а комбинированные - осуществляют сразу несколько видов преобразований (вращение и др.).

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

Новые координаты определяются соотношением

𝑋=∙𝑥

𝑌=∙𝑦

Где и – коэффициенты масштабирования по осям Ox и Oy соответственно.

Отображение - видовое преобразование при котором объект отображается относительно оси Х или 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 , называемой вектор-строка или матрицы  , называемой вектор-столбец. Данные матрицы часто называют координатными векторами. Если геометрическое преобразование представить в виде матрицы, то результат преобразования точки можно представить следующей формулой:

(1),

где M – 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:

**Преобразование масштабирования**

Предположим , тогда

Таким образом

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

Если 0 < = <1 имеет место сжатие, в случае = > 1 - расширение.

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

**Преобразование поворота**

При отрицательных значениях коэффициентов в матрице масштабирования, вместе с масштабированием, происходит отображение объектов относительно различных осей.

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; **α** - угол между осью x и начальным положением отрезка ОР; **θ** - угол, на который выполняется поворот. Определим начальное положение точки Р и ее положение после поворота Р’.

(2)

Применив формулы косинуса и синуса суммы, получим:

Раскрывая скобки и учитывая формулу (2) получаем:

т.е. точка после поворота имеет координаты

Переходя к матричному представлению, получаем матрицу преобразования поворота на произвольный угол **θ** 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 язык программирования C#, который является языком высокого уровня и позволяет быстро и эффективно создавать приложения.

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

Достоинства C#:

1. C# создавался параллельно с 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 их лучшие черты. Общий с этими языками синтаксис, знакомые операторы языка облегчают переход программистов от С++ к C#;
5. Сохранив основные черты своего 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, 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#;
8. Реализация, сочетающая построение надежного и эффективного кода, является немаловажным фактором, способствующим успеху C#.



**1.3 Задание базовой фигуры**

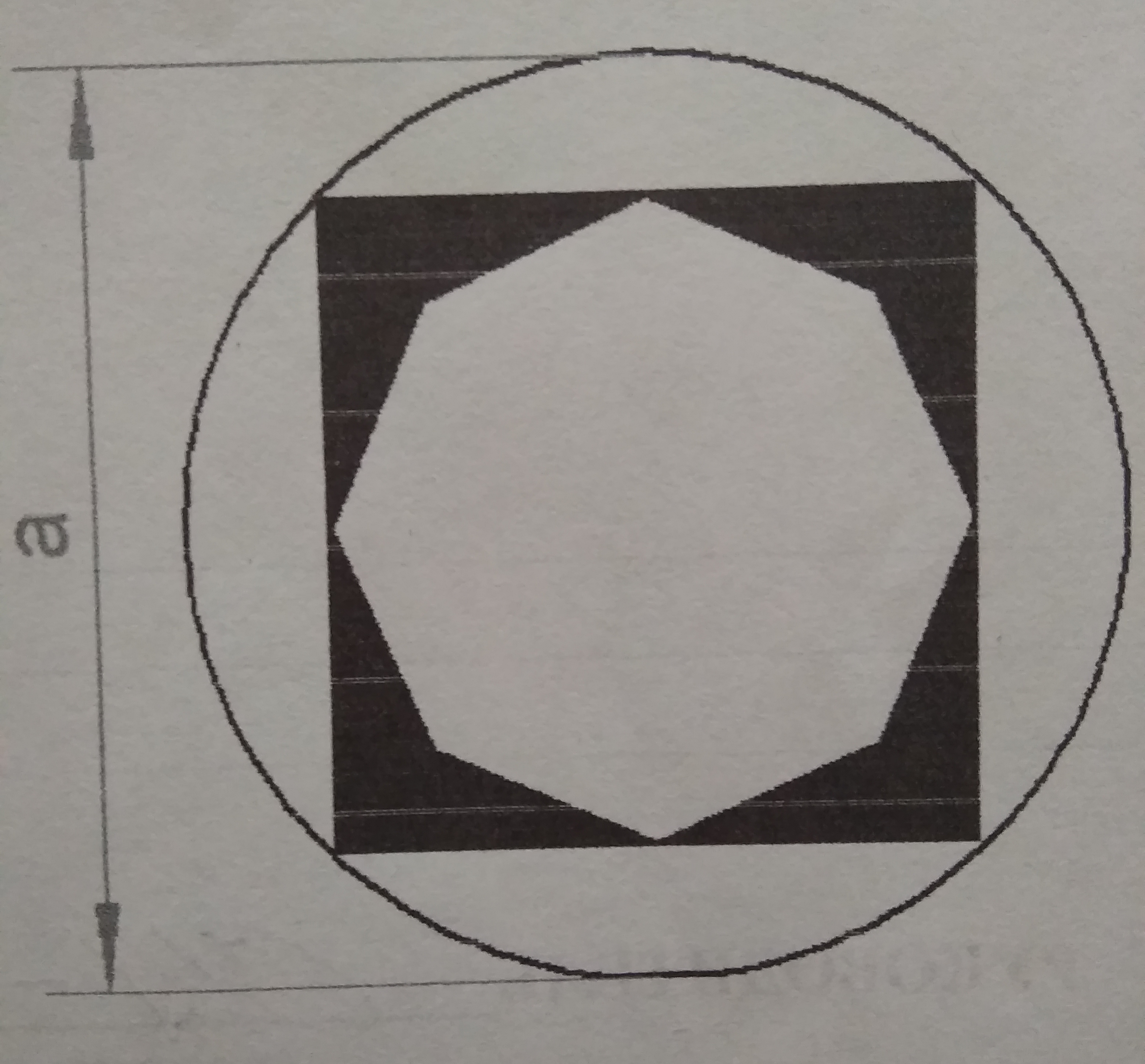


Рисунок 1.1 – Исходный графический объект

Исходный графический объект (рисунок 1.1) представляет собой круг, и квадрат в который вписан восьмигранник. Объект обладает размером стороны круга, величиной a.

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

Отрисовка задней части фигуры реализована полигоном в форме круга. Отрисовка между передней и задней частями фигуры реализована полигоном в форме квадрата. Отрисовка передней части фигуры восьмигранника .

g.FillEllipse(br1, Convert.ToInt32(xc - (a / 2)), Convert.ToInt32(yc - (a / 2)), a, a);

g.FillPolygon(br2, square);

g.FillPolygon(br1, unknown);

Для базовой фигуры точки выражаются через координаты центра и определяющей размер переменной a.

; ;

;

;

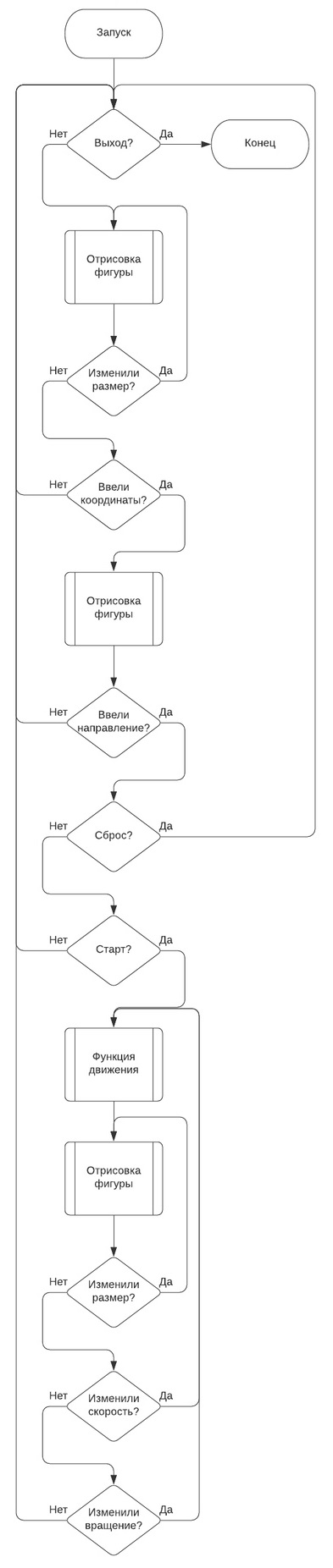
Чтобы изменить цвет фигуры при движении по траектории, был проведен анализ, как должны изменяться цветовые координаты R, G, B в процессе движения. Круг и восьмигранник в начале траектории была синего зеленые (R = 0; G = 128; B = 0), в конце –желтые (R = 255; G = 255; B = 0). Координаты B при движении фигуры возрастающей от 128 до 0. Квадрат в начале траектории были красного цвета (R = 255; G = 0; B = 0), в конце – синего (R = 0; G = 0; B = 255). Координаты R и G при движении фигуры убывают от 255 до 0, а координаты B возрастают от 0 до 255.



2. **РАЗРАБОТКА АЛГОРИТМА РАБОТЫ ПРОГРАММЫ**

Программа начинает свою работу. Производится задание значений, после чего 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: pictureBox, timer, buttondepict, buttonstir, button1, buttonStop, buttondefault, trackBarsize, trackBarspeed, trackBarRotationSpeed, textBoxStartingX, textBoxStartingY, textBoxStartingDirectionX, textBoxStartingDirectionY, labelErrors

pictureBox – область для отрисовки и движения фигуры

timer – таймер запуска процесса движения

buttondepict, – кнопка отрисовки фигуры

buttonstir– кнопка запуска фигуры

button1– кнопка ввода координат направления

buttonStop – кнопка остановки фигуры

buttondefault – кнопка сброса настроек по умолчанию

trackBarsize – ползунок изменения размера

trackBarspeed – ползунок изменения скорости движения

trackBarrotation – ползунок изменения скорости вращения

textBoxStartingX – поле начальных координат X

textBoxStartingY – поле начальных координат Y

textBoxStartingDirectionX – поле начального направления координат X

textBoxStartingDirectionY – поле начального направления координат Y

ErrorText – отображение текста с ошибками в программе

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

{

public partial class FormFigure : Form

{

const int default\_size = 40;

int a;

int v = 1;

int s = 1;

int dirR = 1;

int dirX = 1;

int dirY = 1;

double xc, yc;

double dx = 0.1;

double dy = 0.1;

double fi;

double c\_fi = 0;

bool checkX, checkY;

Point[] square = new Point[4];

Point[] unknown = new Point[8];

double[] x = new double[12];

double[] y = new double[12];

public FormFigure()

{

InitializeComponent();

}

private void FormFigure\_Load(object sender, EventArgs e)

{

a = default\_size;

xc = Convert.ToInt32(a \* Math.Sqrt(2) / 2);

yc = Convert.ToInt32(pictureBox.Height - (a \* Math.Sqrt(2) / 2));

trackBarsize.Value = a;

trackBarspeed.Value = v;

trackBarrotation.Value = s;

}

private double[] x\_rot(double[] x0, double[] y0)

{

double[] x1 = new double[x.Length];

for (int i = 0; i < x.Length; i++)

{

x1[i] = (x0[i]) \* Math.Cos(c\_fi) - (y0[i]) \* Math.Sin(c\_fi);

}

return x1;

}

private double[] y\_rot(double[] x0, double[] y0)

{

double[] y1 = new double[x.Length];

for (int i = 0; i < x.Length; i++)

{

y1[i] = (y0[i]) \* Math.Cos(c\_fi) + (x0[i]) \* Math.Sin(c\_fi);

}

return y1;

}

private byte colorInc()

{

try

{

double col = 225 \* (xc - (a / 2) \* Math.Cos(Math.PI / 6)) /

(pictureBox.Width - a \* Math.Cos(Math.PI / 6));

return Convert.ToByte(col);

}

catch

{

return 0;

}

}

private byte colorDec()

{

try

{

double col = -225 \* ((xc - (a / 2) \* Math.Cos(Math.PI / 6)) /

(pictureBox.Width - a \* Math.Cos(Math.PI / 6))) + 225;

return Convert.ToByte(col);

}

catch

{

return 0;

}

}

private void trackBarsize\_Scroll(object sender, EventArgs e)

{

a = trackBarsize.Value;

if (!timer.Enabled)

{

xc = Convert.ToInt32(a \* Math.Sqrt(2) / 2);

yc = Convert.ToInt32(pictureBox.Height - (a \* Math.Sqrt(2) / 2));

ris();

}

}

private void trackBarspeed\_Scroll(object sender, EventArgs e)

{

v = trackBarspeed.Value;

}

private void trackBarRotationSpeed\_Scroll(object sender, EventArgs e)

{

s = trackBarrotation.Value;

}

**{**



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

{

timer.Stop();

xc = Convert.ToInt32(a \* Math.Sqrt(2) / 2);

yc = Convert.ToInt32(pictureBox.Height - (a \* Math.Sqrt(2) / 2));

dx = 0.1;

dy = 0.1;

c\_fi = 0;

textBoxStartingX.Clear();

textBoxStartingY.Clear();

textBoxStartingEnterCoordX.Clear();

textBoxStartingEnterCoordY.Clear();

ris(); }

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

{

ris();

}

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

{

try

{

double hypotenuse = Math.Pow((Convert.ToInt32(textBoxStartingEnterCoordX.Text) - xc), 2) + Math.Pow((Convert.ToInt32(textBoxStartingEnterCoordY.Text) - yc), 2);

dx = Math.Abs((Convert.ToInt32(textBoxStartingEnterCoordX.Text) - xc)) / Math.Sqrt(hypotenuse);

dy = Math.Abs((Convert.ToInt32(textBoxStartingEnterCoordY.Text) - yc)) / Math.Sqrt(hypotenuse);

if (Convert.ToInt32(textBoxStartingEnterCoordX.Text) > xc)

{

dirX = 1;

checkX = true;

}

else

{

dirX = -1;

checkX = false;

}

if (Convert.ToInt32(textBoxStartingEnterCoordY.Text) > yc)

{

dirY = 1;

checkY = true;

}

else

{

dirY = -1;

checkY = false;

}

}

catch

{

}

}

**private** void textBoxStartingDirectionX\_TextChanged**(object** sender**,** EventArgs e**)**

**{**

**try**

**{**

**if** **(**Convert**.**ToInt32**(**textBoxStartingX**.**Text**)** **<** Convert**.**ToInt32**(**a **\*** Math**.**Sqrt**(**2**)** **/** 2**)** **||** Convert**.**ToInt32**(**textBoxStartingX**.**Text**)** **>** pictureBoxFigure**.**Width**)**

**{**

labelErrors**.**Text **=** String**.**Format**(**"Ошибка в ''X'' находится за координатной плоскостью."**);**

**}**

**else**

**{**

labelErrors**.**Text **=** String**.**Format**(**"Ошибок не обнаружено."**);**

 **}**

**}**

**catch**

**{**

**}**

**}**

private void textBoxStartingX\_TextChanged(object sender, EventArgs e)

{

try

{

if (Convert.ToInt32(textBoxStartingX.Text) < Convert.ToInt32(a \* Math.Sqrt(2) / 2) || Convert.ToInt32(textBoxStartingX.Text) + Convert.ToInt32(a \* Math.Sqrt(2) / 2) > pictureBox.Width)

{

xc = Convert.ToInt32(a \* Math.Sqrt(2) / 2);

ErrorText.Text = String.Format("Ошибка в ''X'' находится за координатной плоскостью.");

}

else

{

xc = Convert.ToInt32(textBoxStartingX.Text);

ErrorText.Text = String.Format("Ошибок нет...");

}

}

catch

{

}

}

private void textBoxStartingY\_TextChanged(object sender, EventArgs e)

{

try

{

if (Convert.ToInt32(textBoxStartingY.Text) < Convert.ToInt32(a \* Math.Sqrt(2) / 2) || Convert.ToInt32(textBoxStartingY.Text) + Convert.ToInt32(a \* Math.Sqrt(2) / 2) > pictureBox.Height)

{

yc = Convert.ToInt32(pictureBox.Height - (a \* Math.Sqrt(2) / 2));

ErrorText.Text = String.Format("Ошибка в ''Y'' находится за координатной плоскостью.");

}

else

{

yc = Convert.ToInt32(textBoxStartingY.Text);

ErrorText.Text = String.Format("Ошибок нет...");

}

}

catch

{

}

}

private void textBoxStartingEnterCoordX\_TextChanged(object sender, EventArgs e)

{

try

{

if (Convert.ToInt32(textBoxStartingX.Text) < Convert.ToInt32(a \* Math.Sqrt(2) / 2) || Convert.ToInt32(textBoxStartingX.Text) > pictureBox.Width)

{

ErrorText.Text = String.Format("Ошибка в ''X'' находится за координатной плоскостью.");

}

else

{

ErrorText.Text = String.Format("Ошибок нет...");

}

}

catch

{

}

 }

private void textBoxStartingEnterCoordY\_TextChanged(object sender, EventArgs e)

{

try

{

if (Convert.ToInt32(textBoxStartingY.Text) < Convert.ToInt32(a \* Math.Sqrt(2) / 2) || Convert.ToInt32(textBoxStartingY.Text) > pictureBox.Height)

{

ErrorText.Text = String.Format("Ошибка в ''X'' находится за координатной плоскостью.");

}

else

{

ErrorText.Text = String.Format("Ошибок нет...");

}

}

catch

{

}

}

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

{

if (dx != 0.1 && dy != 0.1)

{

timer.Start();

ErrorText.Text = String.Format("Ошибок нет...");

}

else

{

ErrorText.Text = String.Format("Направление не задано.");

}

}

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

{

}

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

{

}

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

{

}

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

{

if (x.Max() + xc > pictureBox.Width && checkX)

{

dirX = -1;

checkX = false;

if (dirY == -1)

{

dirY = 1;

checkY = true;

}

else

{

dirY = -1;

checkY = false;

}

if (dirR == 1) dirR = -1;

else dirR = 1;

}



if (x.Min() + xc < 0 && !checkX)

{

dirX = 1;

checkX = true;

if (dirY == -1)

{

dirY = 1;

checkY = true;

}

else

{

dirY = -1;

checkY = false;

}

if (dirR == 1) dirR = -1;

else dirR = 1;

}

if (y.Max() + yc > pictureBox.Height && checkY)

{

dirY = -1;

checkY = false;

if (dirX == -1)

{

dirX = 1;

checkX = true;

}

else

{

dirX = -1;

checkX = false;

}

if (dirR == 1) dirR = -1;

else dirR = 1;

}

if (y.Min() + yc < 0 && !checkY)

{

dirY = 1;

checkY = true;

if (dirX == -1)

{

dirX = 1;

checkX = true;

}

else

{

dirX = -1;

checkX = false;

}

if (dirR == 1) dirR = -1;

else dirR = 1;

}

c\_fi += fi \* s;

fi = dirR \* Math.PI / (180);

xc = xc + dx \* dirX \* v;

yc = yc + dy \* dirY \* v;

ris();

}

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

{

a = default\_size;

trackBarsize.Value = default\_size;

if (!timer.Enabled)

{

xc = Convert.ToInt32(a \* Math.Sqrt(2) / 2);

yc = Convert.ToInt32(pictureBox.Height - (a \* Math.Sqrt(2) / 2));

ris();

}

}

private void ris()

{

Graphics g = pictureBox.CreateGraphics();

Brush br1 = new SolidBrush(Color.FromArgb(colorInc(), 128, 0));

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

g.Clear(SystemColors.Control);

x[0] = Convert.ToInt32(-(a / 2) \* Math.Cos(Math.PI / 4));

y[0] = Convert.ToInt32(-(a / 2) \* Math.Sin(Math.PI / 4));

x[1] = Convert.ToInt32(+(a / 2) \* Math.Cos(Math.PI / 4));

y[1] = Convert.ToInt32(-(a / 2) \* Math.Sin(Math.PI / 4));

x[2] = Convert.ToInt32(+(a / 2) \* Math.Cos(Math.PI / 4));

y[2] = Convert.ToInt32(+(a / 2) \* Math.Sin(Math.PI / 4));

x[3] = Convert.ToInt32(-(a / 2) \* Math.Cos(Math.PI / 4));

y[3] = Convert.ToInt32(+(a / 2) \* Math.Sin(Math.PI / 4));

//

x[4] = Convert.ToInt32(0);

y[4] = Convert.ToInt32(-((a / Math.Sqrt(2)) / 2));

x[5] = Convert.ToInt32(+((a / Math.Sqrt(2)) / 2) \* Math.Cos(Math.PI / 4));

y[5] = Convert.ToInt32(-((a / Math.Sqrt(2)) / 2) \* Math.Sin(Math.PI / 4));

x[6] = Convert.ToInt32(+((a / Math.Sqrt(2)) / 2));

y[6] = Convert.ToInt32(0);

x[7] = Convert.ToInt32(+((a / Math.Sqrt(2)) / 2) \* Math.Cos(Math.PI / 4));

y[7] = Convert.ToInt32(+((a / Math.Sqrt(2)) / 2) \* Math.Sin(Math.PI / 4));

x[8] = Convert.ToInt32(0);

y[8] = Convert.ToInt32(+((a / Math.Sqrt(2)) / 2));

x[9] = Convert.ToInt32(-((a / Math.Sqrt(2)) / 2) \* Math.Cos(Math.PI / 4));

y[9] = Convert.ToInt32(+((a / Math.Sqrt(2)) / 2) \* Math.Sin(Math.PI / 4));

x[10] = Convert.ToInt32(-((a / Math.Sqrt(2)) / 2));

y[10] = Convert.ToInt32(0);

x[11] = Convert.ToInt32(-((a / Math.Sqrt(2)) / 2) \* Math.Cos(Math.PI / 4));

y[11] = Convert.ToInt32(-((a / Math.Sqrt(2)) / 2) \* Math.Sin(Math.PI / 4));

double[] temp = new double[16];

temp = x;

x = x\_rot(temp, y);

y = y\_rot(temp, y);

square[0] = new Point((int)(x[0] + xc), (int)(y[0] + yc));

square[1] = new Point((int)(x[1] + xc), (int)(y[1] + yc));

**** square[2] = new Point((int)(x[2] + xc), (int)(y[2] + yc));

square[3] = new Point((int)(x[3] + xc), (int)(y[3] + yc));

unknown[0] = new Point((int)(x[4] + xc), (int)(y[4] + yc));

unknown[1] = new Point((int)(x[5] + xc), (int)(y[5] + yc));

unknown[2] = new Point((int)(x[6] + xc), (int)(y[6] + yc));

unknown[3] = new Point((int)(x[7] + xc), (int)(y[7] + yc));

unknown[4] = new Point((int)(x[8] + xc), (int)(y[8] + yc));

unknown[5] = new Point((int)(x[9] + xc), (int)(y[9] + yc));

unknown[6] = new Point((int)(x[10] + xc), (int)(y[10] + yc));

unknown[7] = new Point((int)(x[11] + xc), (int)(y[11] + yc));

g.FillEllipse(br1, Convert.ToInt32(xc - (a / 2)), Convert.ToInt32(yc - (a / 2)), a, a);

g.FillPolygon(br2, square);

g.FillPolygon(br1, unknown);

}

}

}



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

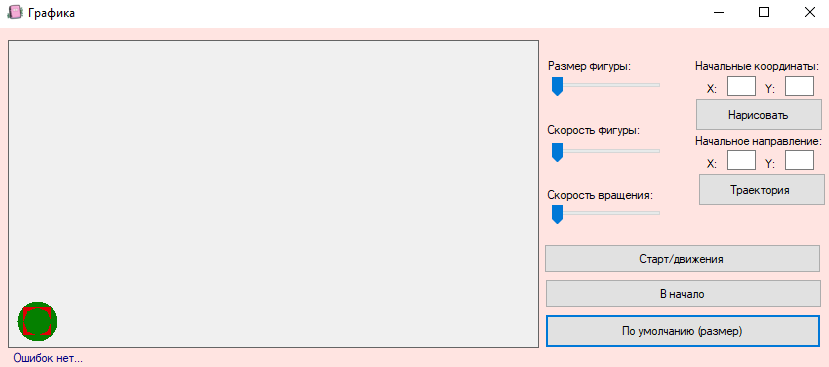


Рисунок 5.1 – Рабочее окно программы

1. На рисунке 5.2 изображено масштабирование объекта



Рисунок 5.2 ­– Рабочее окно программы с измененным размером фигуры

1. На рисунке 5.3 показан процесс вращения фигуры

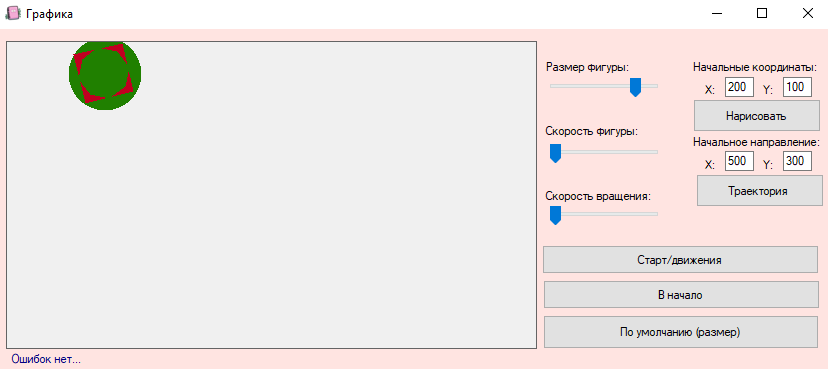


Рисунок 5.3 – Рабочее окно программы с измененным размером и

скоростью вращения фигуры

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

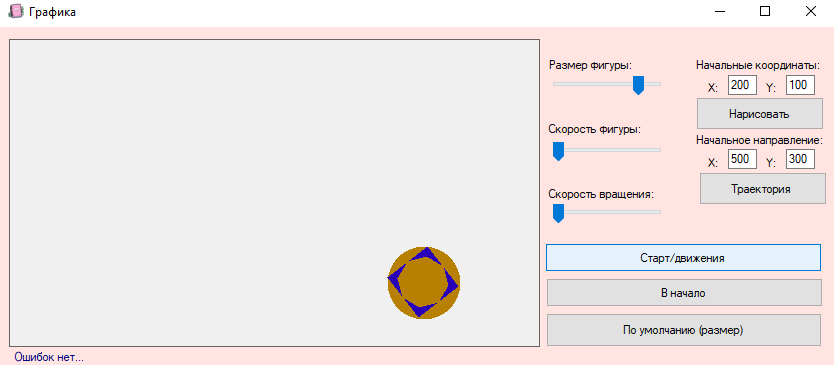


Рисунок 5.4 – ­­Рабочее окно программы, показывающее изменение

периода и цвета фигуры от координат



**Заключение**

В результате данного курсового проекта было разработано Windows-приложение, которое непосредственно показывает возможности создания и преобразования графических объектов с помощью языка программирования 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. Викторова, Т. С. Системы компьютерной графики / Т. С. Викторова, С. Д. Парфенов. – Вязьма: филиал ФГБОУ ВПО «МГИУ» в г. Вязьме, 2012. – 165 с.
4. Боресков, А.В. Компьютерная графика: Учебник и практикум для прикладного бакалавриата / А.В. Боресков, Е.В. Шикин. - Люберцы: Юрайт, 2016. - 219 c.
5. Visual C# 2008 Базовый курс/К. Уотсон [и др.]; под общ. ред. К. Нейгеля. – издательский дом Вильямс, 2009. – 1216 с.
6. С# 2008 и платформа .NET 3.5 для профессионалов/ К. Нейгел [и др.]; под общ. ред. Кристиан Нейгеля. – издательский дом Вильямс, 2008. – 1392 с.