Министерство образования Республики Беларусь

Белорусский Национальный Технический Университет

Факультет информационных технологий и робототехники

Кафедра программного обеспечения информационных

систем и технологий

**КУРСОВАЯ РАБОТА**

по дисциплине «**Разработка приложений в визуальных средах**»

на тему: «**Разработка приложения формирования и отображения звезд различных размеров**»

Выполнил**:** ст. гр. 10701320 Акимов Е.С.

Принял: доц. Гурский Н.Н.

Минск 2022

БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Кафедра программного обеспечения информационных систем и технологий

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**к курсовой работе**

**по дисциплине «**Разработка приложений в визуальных средах»

Тема: «Разработка приложения формирования и отображения звезд различных размеров»

**Исполнитель**: Акимов Е.С.

(подпись)

**Студент 2 курса 10701320 группы**

**Руководитель**: доц. Гурский Н.Н.

(подпись)

Минск 2022

ОГЛАВЛЕНИЕ

[ВВЕДЕНИЕ 4](#_Toc103198210)

[1. МАТЕМАТИЧЕСКАЯ ФОРМУЛИРОВКА ЗАДАЧИ 5](#_Toc103198211)

[2. ОПИСАНИЕ ПРОГРАММЫ 6](#_Toc103198212)

[2.1 Структурная схема программы 6](#_Toc103198213)

[2.2 Описание разработанного класса 6](#_Toc103198214)

[2.3 Основные возможности программы 7](#_Toc103198215)

[3. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 8](#_Toc103198216)

[4. МЕТОДИКА ИСПЫТАНИЙ 15](#_Toc103198217)

[ЗАКЛЮЧЕНИЕ 18](#_Toc103198218)

[СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ 19](#_Toc103198219)

[Листинг исходных кодов программ 20](#_Toc103198220)

# ВВЕДЕНИЕ

Компьютерная модель - компьютерная программа, работающая на отдельном компьютере, суперкомпьютере или множестве взаимодействующих компьютеров (вычислительных узлов), реализующая абстрактную, то есть информационную модель некоторой системы. Компьютерные модели стали обычным инструментом численно-математического моделирования и применяются в физике, астрофизике, механике, химии, биологии, экономике, социологии, метеорологии, других науках и прикладных задачах в различных областях радиоэлектроники, машиностроения, автомобилестроения и проч. Компьютерные модели используются для получения новых знаний о моделируемом объекте или для приближенной оценки поведения систем, слишком сложных для логико-аналитического исследования.

Компьютерное моделирование является одним из эффективных методов изучения сложных систем. Компьютерные модели проще и удобнее исследовать в силу их возможности проводить так называемые "вычислительные эксперименты", которые на самом деле экспериментами не являются, так как информация берется не из физического эксперимента, физической реальности, а из модельного представления о ней, проводят в тех случаях, когда реальные эксперименты затруднены из-за финансовых или физических препятствий, или могут дать непредсказуемо опасный результат. В случае корректной логики и корректной формализации на этапе создания компьютерных моделей имеется возможность выявить основные факторы, определяющие количественные свойства изучаемого объекта-оригинала (или целого класса объектов), в частности, исследовать отклик моделируемой физической системы на изменения ее параметров и начальных условий.

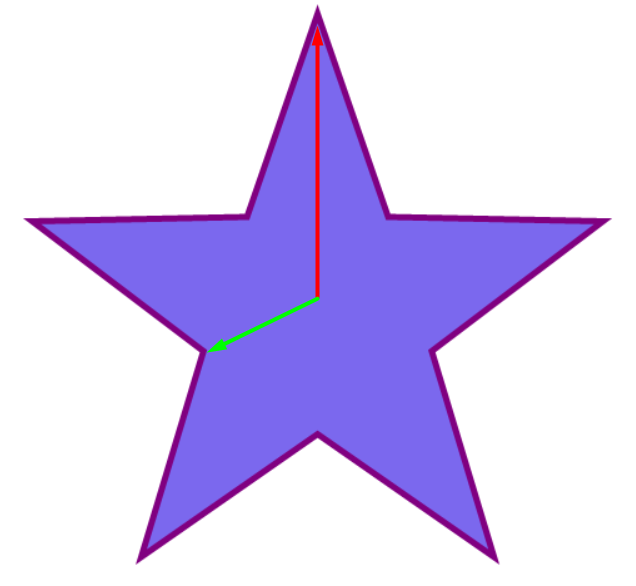
Построение компьютерной модели базируется на абстрагировании от конкретной природы явлений или изучаемого объекта-оригинала и состоит из двух этапов — сначала создание качественно-логической, а затем и количественно-математической модели. Компьютерное же моделирование заключается в проведении серии "вычислительных экспериментов" на компьютере, целью которых является анализ на внутреннюю непротиворечивость модели и получение количественных данных о процессе функционирования модели.

Затем исследователь производит интерпретацию, то есть объяснение этих количественных результатов и их содержательное, то есть неформальное сопоставление с реальным поведением изучаемого объекта, а также частое и многократное последующее уточнение модели и т. д.

Целью курсовой работы является разработка приложения формирования и отображения звезд различных размеров как в 2d, так и в 3d. Для работы был выбран язык программирования C# (WinForms), который даёт возможность быстрого создания приложений с графическим интерфейсом, а также позволяет реализовать концепции ООП. Также использовалась библиотека OpenGl.

# МАТЕМАТИЧЕСКАЯ ФОРМУЛИРОВКА ЗАДАЧИ

Дана размер звезды, который будет являться внешним радиусом звезды (см. рисунок 1.1).



Rвнешн

Rвн

Рисунок 1.1 – Система из двух маятников

Расчёт точек углов звезд будет происходить по формулам:

|  |  |
| --- | --- |
|  | (1) |
|  | (2) |

где n – номер точки (угла), считая по часовому ходу, R = внутреннему радиусу, если n не кратно двум, и R = внешнему радиусу, если n кратно двум.

# ОПИСАНИЕ ПРОГРАММЫ

# 2.1 Структурная схема программы

Структурно программа состоит из главной формы и разработанных классов и приведена на (см. Рисунок 2.1).

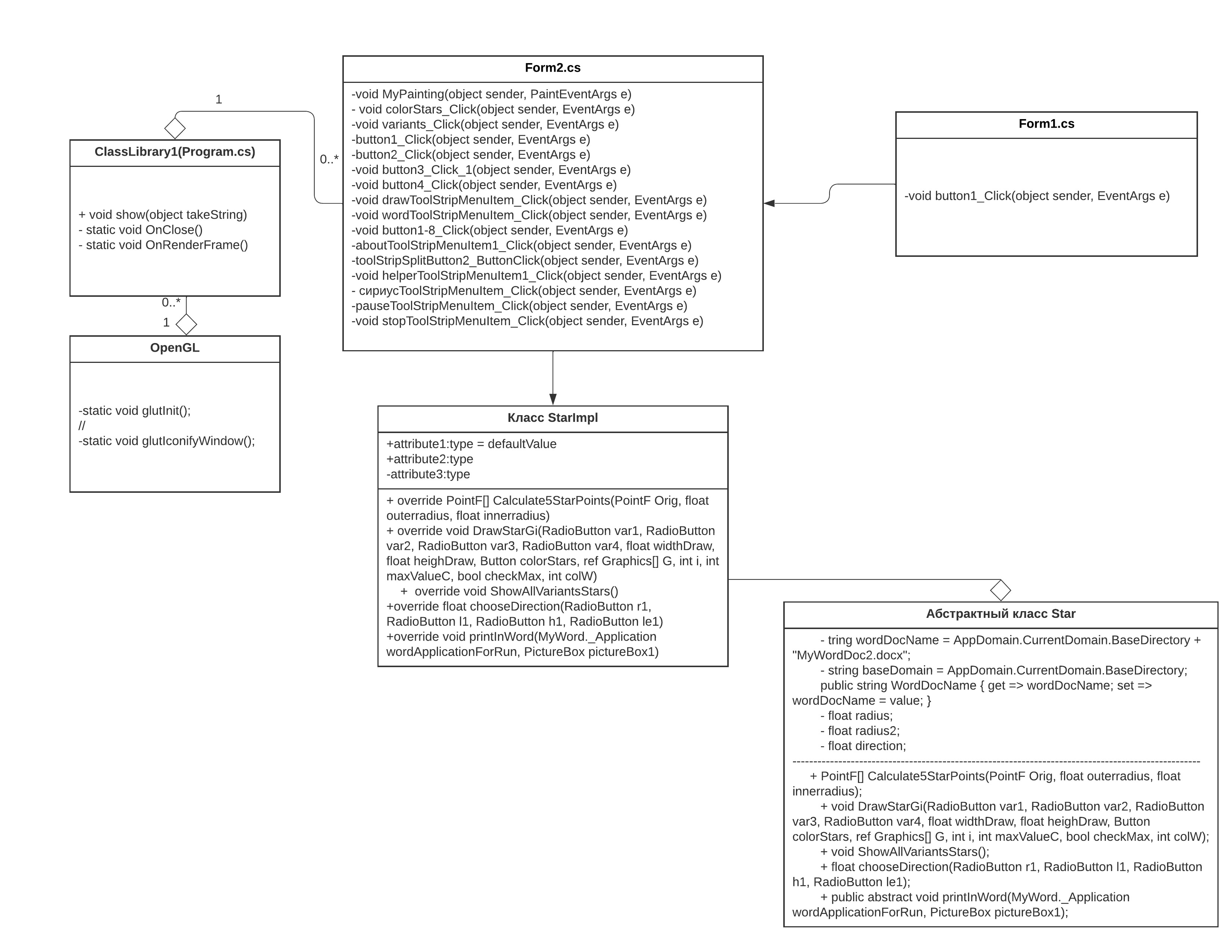


Рисунок 2.1 Структурная схема связей модулей программы.

# 2.2 Описание разработанного класса

В результате анализа предметной области для её описания и моделирования были разработаны:

class Star – абстрактный класс отвечающий за необходимые методы, который должен содержать расширяющий его класс.

class StarImpl – класс расширяющий класс Star.

public override PointF[] Calculate5StarPoints(PointF Orig, float outerradius, float innerradius) – метод отвечающий за расчёт точек концов звёзд в зависимости от передаваемого размера звезды.

public override void DrawStarGi(RadioButton var1, RadioButton var2, RadioButton var3, RadioButton var4, float widthDraw, float heighDraw, Button colorStars, ref Graphics[] G, int i, int maxValueC, bool checkMax, int colW) – метод отвечающий за рисование звёзд в зависимости от количества передаваемых звёзд и их размера на Graphics

public override void ShowAllVariantsStars() – откроет файл-word, где можно посмотреть варианты звёзд.

public override float chooseDirection(RadioButton r1, RadioButton l1, RadioButton h1, RadioButton le1) – отвечает за направление вращения 3d звёзд.

public override void printInWord(MyWord.\_Application wordApplicationForRun, PictureBox pictureBox1) – отвечает за вывод последнего отображения звёзд в word-файл.

class Program – отвечает за отображение Сириуса в 3d модели

public void show(object takeString) – является своего рода main-ом для отображения звзёзд.

private static void OnClose() – отвечает за закрытие 3d модели

private static void OnRenderFrame() – отвечает за вращение звезды.

# 2.3 Основные возможности программы

Сначала нас встречает окно-обёртка с единственной кнопкой «Запустить проект».

Далее нас встречает окно для взаимодействия с программой пользователю. Функционал, который предоставляет форма:

* Вызов «О программе»
* Вызов Helper’a
* Нарисовать 2d звезду
* Открыть последнее рисование в Word
* Показать Сириус в 3D
* Остановить вращение звезды
* Продолжить вращение звезды
* Закрыть вращение звезды и вернуться к 2D рисованию
* Посмотреть варианты звёзд
* Выбрать варианты звёзд
* Выбрать вращение 3d модели
* Выбрать цвет звезды (будет преобладать)
* Выбрать количество звёзд
* Выбрать размер как 2D, так и 3D звезды/Сириуса.
* Остановка программы

# РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

Для запуска приложения необходимо нажать кнопку «Запустить проект» (см. рисунок 3.1), которая вызовет exe файл . После запуска появится рабочее пространство приложения (см. рисунок 3.2).

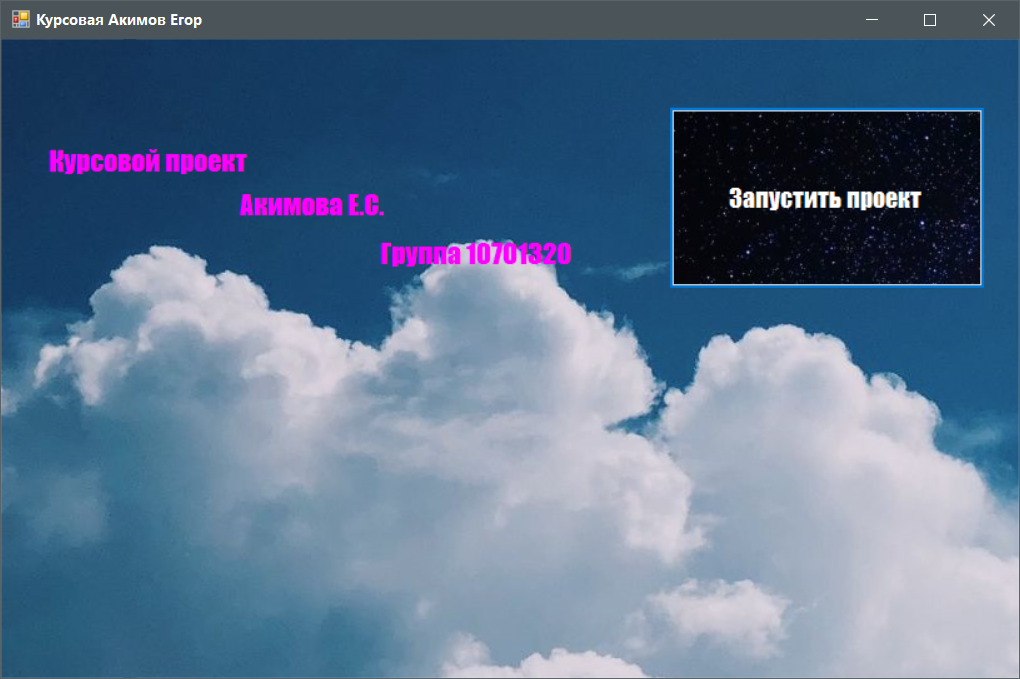


Рисунок 3.1 Главное окно программы

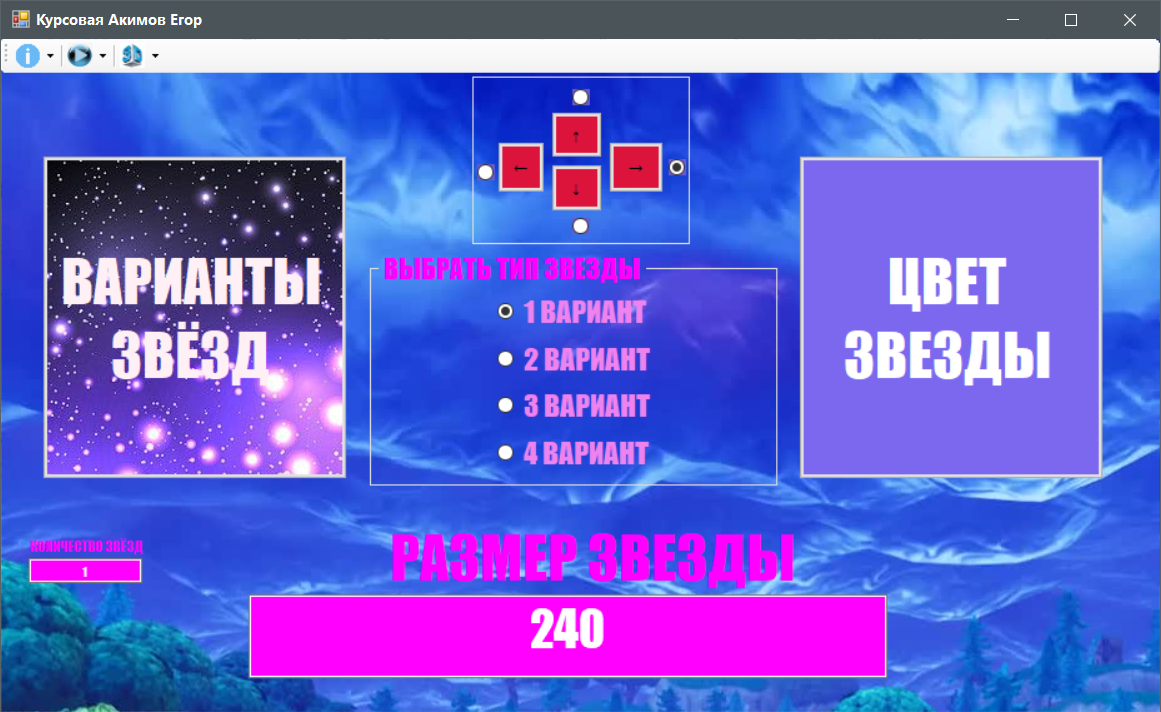


Рисунок 3.2 Рабочая область программы

Для просмотра вариантов звёзд нужно нажать на кнопку «Варианты звёзд» (см. рисунок 3.3)

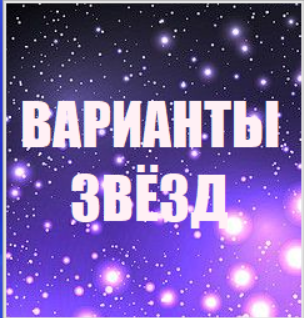


Рисунок 3.3 Варианты звёзд

Для выбора понравившегося варианта звезды ставим галочку напротив нужного варианта (см. рисунок 3.4)



Рисунок 3.4 Выбор варианта звезды

Для выбора преобладающего цвета звезды жмём «Цвет звезды» (см. рисунок 3.5) и выбираем понравившейся цвет.



Рисунок 3.5 Цвет звезды

Для выбора нужного отображения 2d-звезд жмём пишем нужное нам количество звёзд.



Рисунок 3.6 Количество звёзд

Далее выбираем нужный нам размер звезды как для 2D, так и для 3D отображения (см. рисунок 3.7)



Рисунок 3.7 Размер одной звезды

После конфигурации звезды мы жмём на кнопку старта(см. рисунок 3.8)



Рисунок 3.8 Кнопка старта рисвания

Сконфигурённые звезды будут немедленно отображены (см. рисунок 3.9)

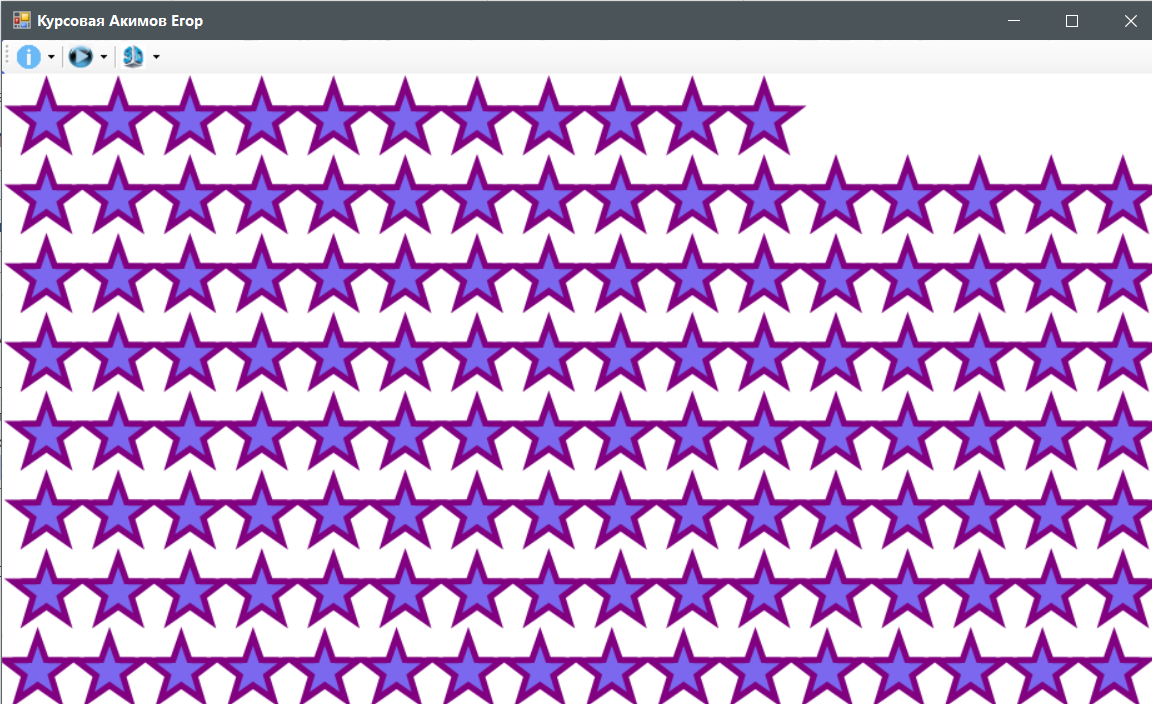


Рисунок 3.9 Результат работы

Также мы можем нажать «Открыть в word» и увидим последнее рисование в word файле (см. рисунок 3.10 и 3.11)

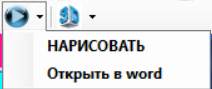


Рисунок 3.10 Кнопка открыть в word

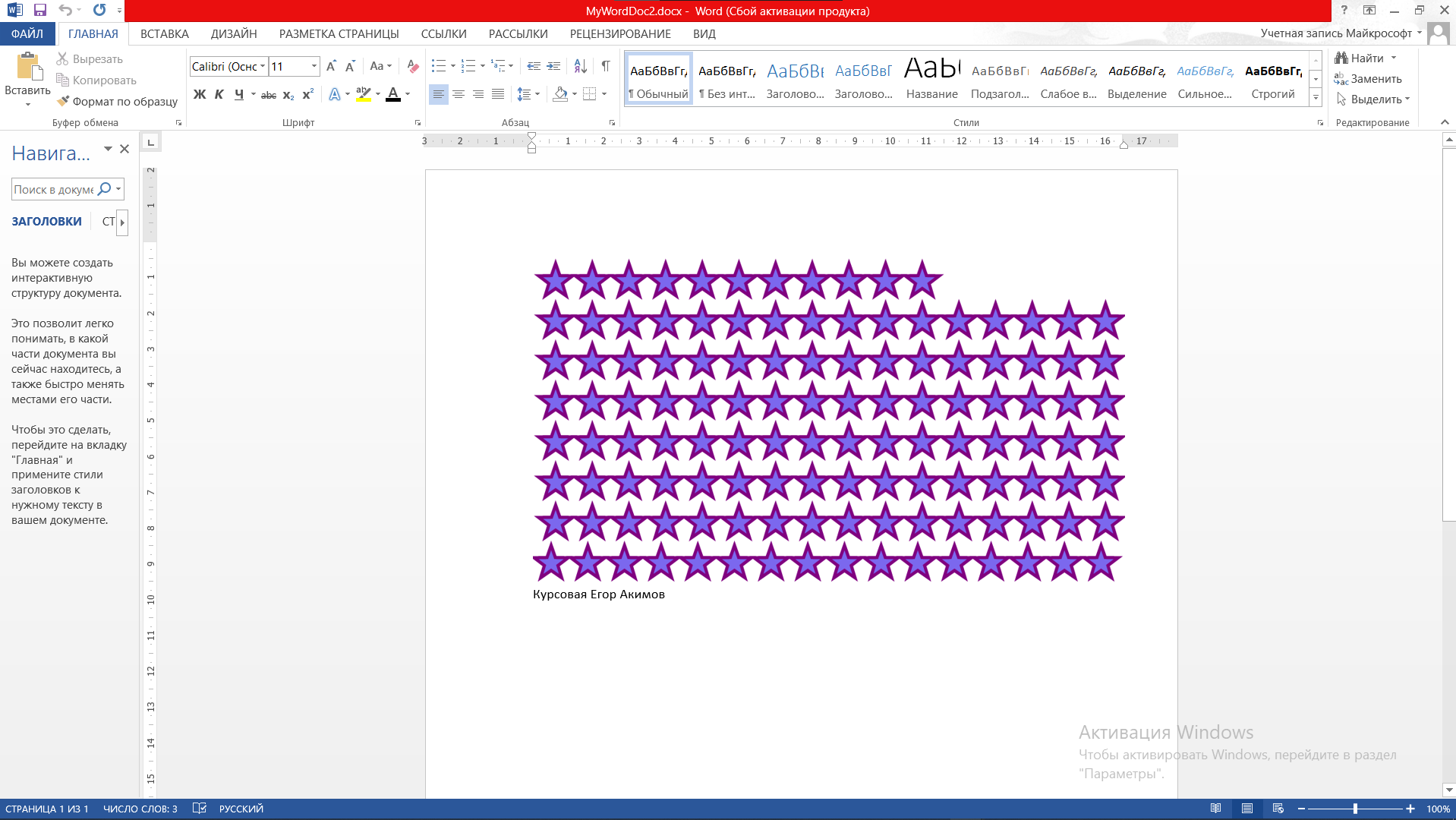


Рисунок 3.11 Результат в word

Далее представлен функционал для отображение 3D Сириуса. Для выбора направления вращения Сириуса нажмём на одну из четырёх кнопок (см. рисунок 3.12)

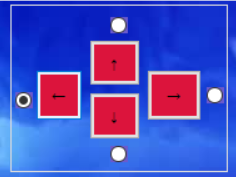


Рисунок 3.12 Направление вращения звезды

Для старта 3D модели нажмём на кнопку «3D» (см. рисунок 3.13)



Рисунок 3.13 Старт 3D модели

3D модель отражена (см. рисунок 3.14)

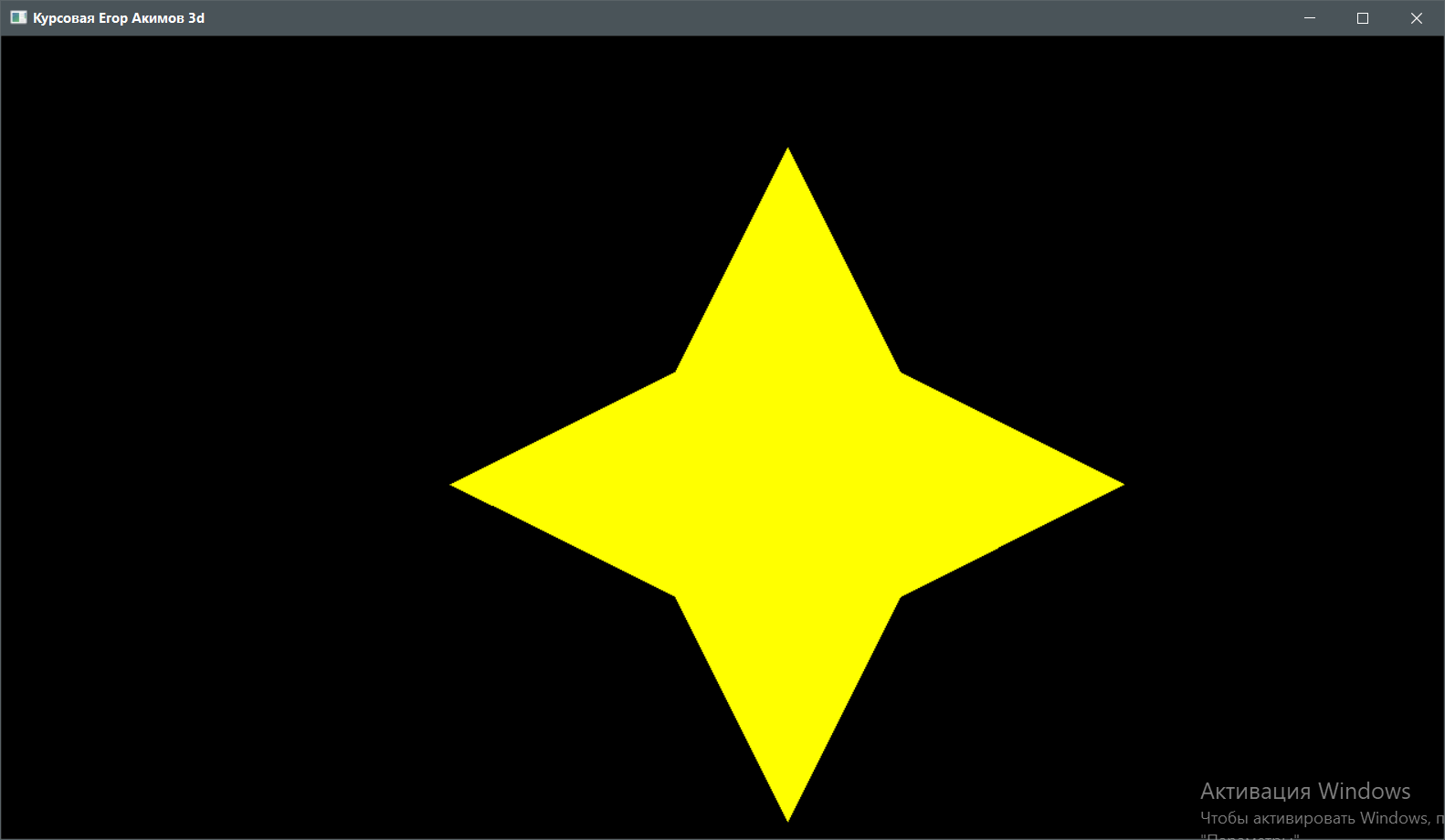


Рисунок 3.14 Вращение 3D-Сириуса

Для временной остановки вращения нажмите «Pause» (см. рисунок 3.15). Для продолжения вращения нажмите «Сириус» (см. рисунок 3.16)

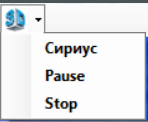


Рисунок 3.15 Остановка вращения 3D-Сириуса

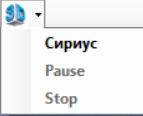


Рисунок 3.16 Продолжения вращения

Для завершения 3D моделирования нажмите «Stop». Программа будет остановлена (см. рисунок 3.17).

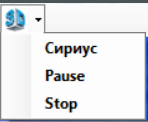


Рисунок 3.17 Кнопка остановки «Stop»

Для просмотра информации «О программе» или же «About» нажмите кнопку «About» (см. рисунок 3.18).

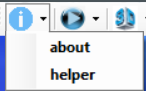


Рисунок 3.18 Кнопка «About»

Для открытия программы помощника или же helper'а нажмите «helper». (см. рисунок 3.19).

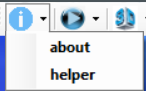


Рисунок 3.19 Кнопка «helper»

# МЕТОДИКА ИСПЫТАНИЙ

Целью проведения испытаний является проверка работоспособности (надежности) программы при различных условиях её функционирования.

Программа должна обеспечивать корректность ввода исходных данных (путем осуществления соответствующих проверок и информирования пользователя о возникших неточностях в работе), а также получение непротиворечивого результата.

Для обеспечения нормальной работы программы требуется наличие необходимых динамических библиотек, а также приложения Word.

Первое испытание будет проведено форме приложения (см. рисунок 4.1).

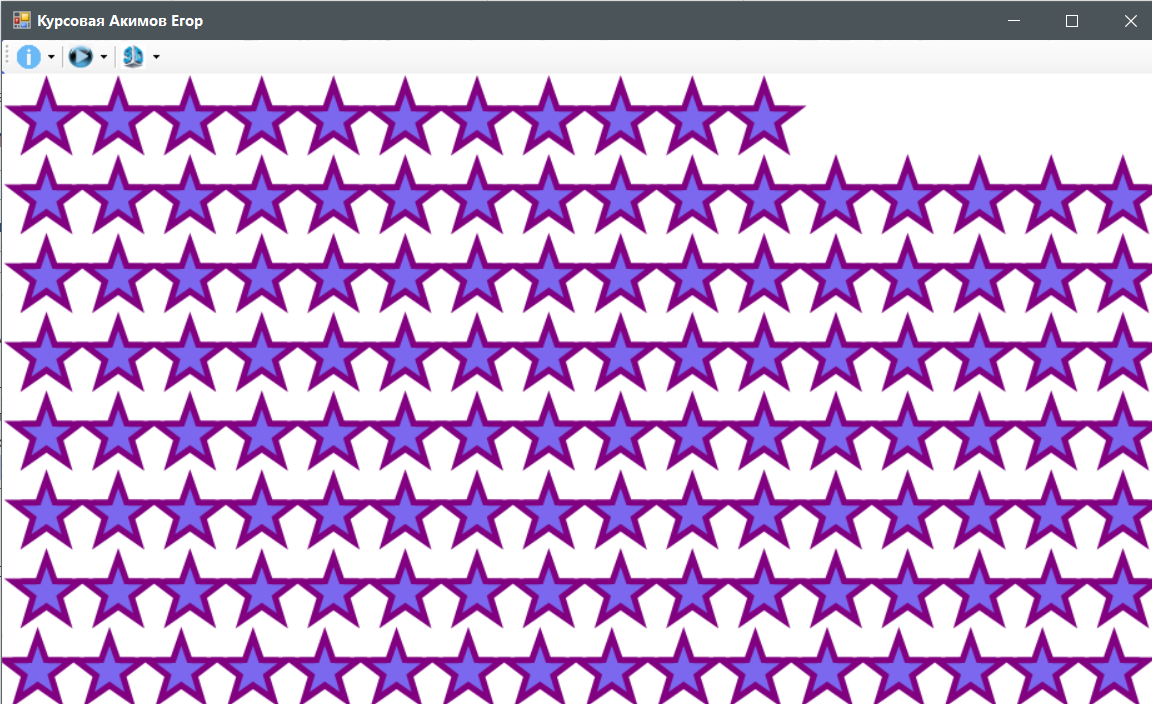


Рисунок 4.1 – Работа при нормальных условиях

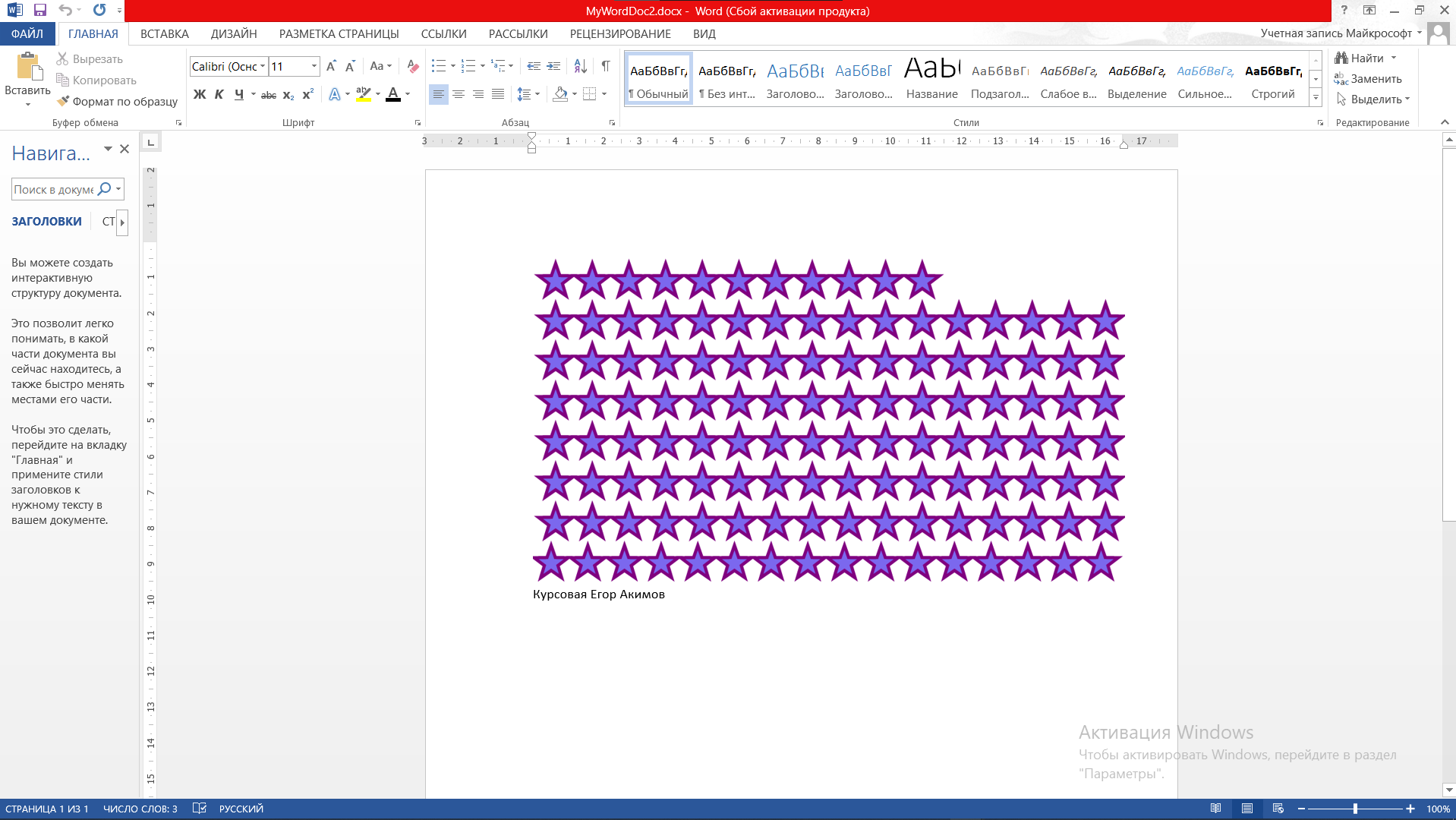


Рисунок 4.2 – Результат вывода screenshot в word

Далее попробуем отобразить «Сириус» в 3D моделировании (см. рисунок 4.3).

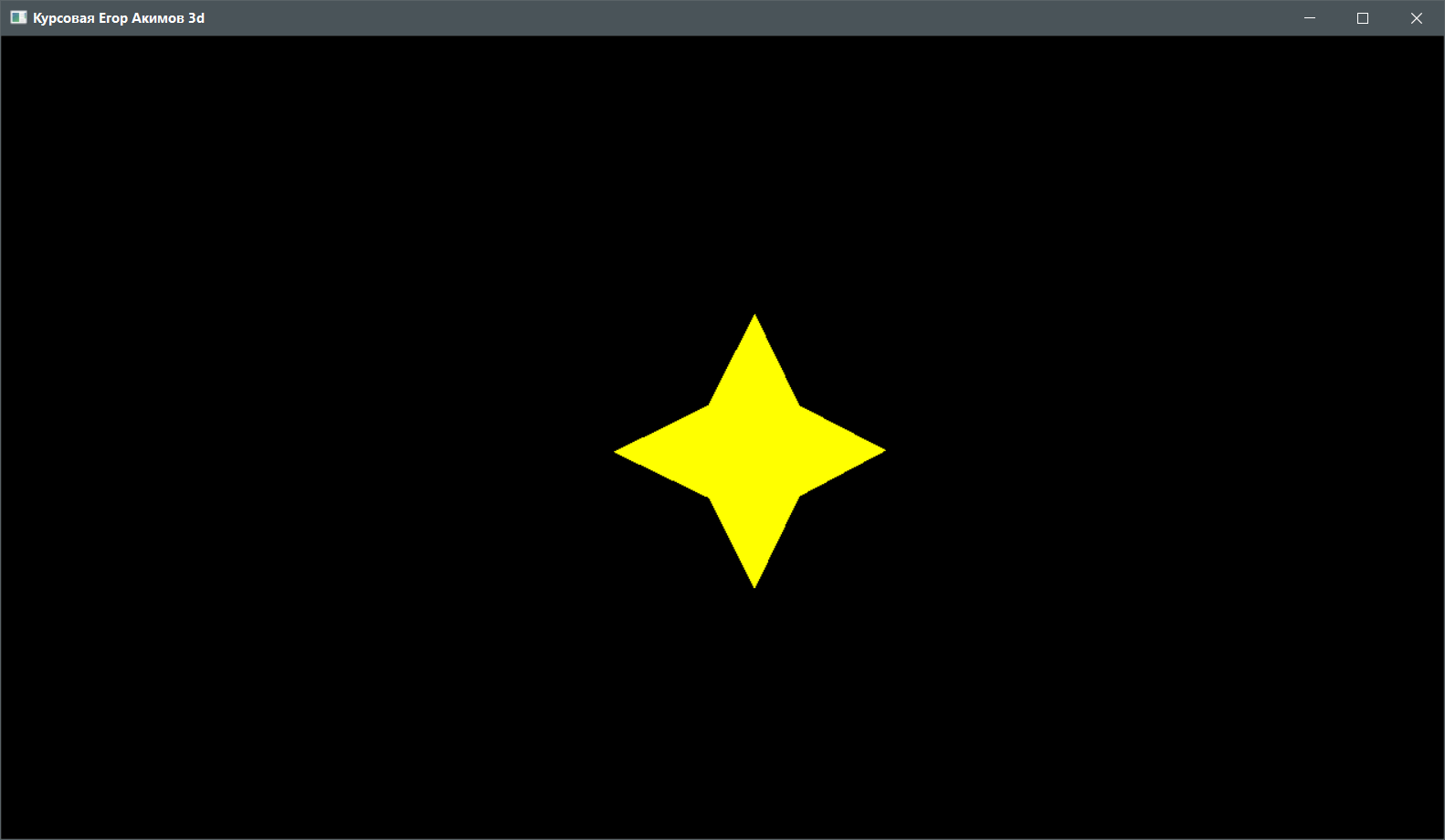


Рисунок 4.3 – Результат отображения в 3D

Далее проведем запуск если звезда/звёзды не буду помещаться в наш экран (см. рисунок 4.4).

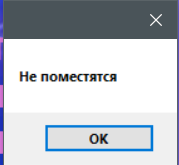


Рисунок 4.4 – Окно предупреждения о том, что программа звезда/звезды не поместятся.

Также попытаемся сделать звезду в 3D больше допустимого значения (см. рисунок 4.5)

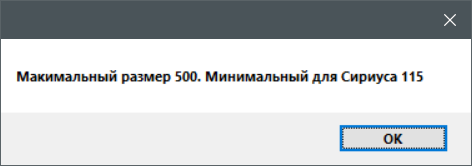


Рисунок 4.5 – Окно предупреждения о том, что документ уже используется другим ресурсом

В результате тестирования приложения ошибок обнаружено не было. Следует считать, что в целом программа протестирована, отвечает поставленным требованием и вполне работоспособна.

# ЗАКЛЮЧЕНИЕ

В ходе выполнения курсовой работы был приобретен опыт при разработке объектно-ориентированных программ и были изучены C#, .NET Framework, Windows Forms, OpenGL, методы симуляции и просчета физических систем, работа с формами, классами, DLL, COM-объектами и офисными приложениями Microsoft.

Разработана программа по формированию и моделированию 2D/3D звёзд с учетом параметров системы, составлена форма для отображения результатов и управления симуляцией. Также программа реализует следующий функционал: отображение звёзд в 2D и 3D формате с выбором её/их размера, отображение большого количества звёзд в 2D формате, загрузка рисования в word

# СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Гринчишин Я.Т. и др. Алгоритмы и программы на Бейсике: Учеб. Пособие для студентов пед. Ин-тов по физ.-мат. Спец./Я. Т. Гринчишин, В. И. Ефимов, А. Н. Ломакович. –М.: Просвещение. 1988.-160с.
2. CLRviaC#. Программирование на платформе Microsoft .NETFramework 4.5 на языке C#. 4-е изд. – СПб.; Питер, 2013 – 896с.
3. С# для чайников.: Пер. с англ. – СПб. : ООО «Диалектика», 2019. – 608 с.

**ПРИЛОЖЕНИЕ A**

# Листинг исходных кодов программ

**Файл Form2.cs**

using System;

using System.Drawing;

using System.Drawing.Drawing2D;

using System.Windows.Forms;

using MyWord = Microsoft.Office.Interop.Word;

using Xceed.Words.NET;

using Xceed.Document.NET;

using Microsoft.Office.Interop.Word;

using System.Threading;

using res2mb.Scripts;

namespace res2mb

{

public partial class Form2 : Form

{

Star star = new StarImpl();

bool mode = false;

float radius;

float radius2;

private PictureBox pictureBox1;

int colW;

int colH;

Thread tred2;

private MyWord.\_Application wordApplicationForRun;

ClassLibrary1.Class1 s = new ClassLibrary1.Class1();

public Form2()

{

InitializeComponent();

tred2 = new Thread(s.show);

}

private void Form2\_Load(object sender, EventArgs e)

{

}

private void MyPainting(object sender, PaintEventArgs e)

{

int maxValueC = int.Parse(maxValue.Text);

Graphics[] G = new Graphics[maxValueC];

for (int z = 0; z < maxValueC; z++)

{

G[z] = e.Graphics;

G[z].SmoothingMode = SmoothingMode.HighQuality;

G[z].Clear(Color.White);

}

bool checkMax = true;

if (maxValueC == 1)

{

checkMax = true;

}

else

{

if (int.Parse(sizeStarsNumber.Text) <= 350)

{

checkMax = true;

}

else

{

checkMax = false;

MessageBox.Show("При значении больше 1 звезды их величина должна быть меньше 350");

draw.Text = "НАРИСОВАТЬ";

typeStars.Visible = true;

sizeStars.Visible = true;

groupBox1.Visible = true;

sizeStarsNumber.Visible = true;

variants.Visible = true;

colorStars.Visible = true;

label1.Visible = true;

maxValue.Visible = true;

button4.Visible = true;

button1.Visible = true;

button3.Visible = true;

h1.Visible = true;

r1.Visible = true;

le1.Visible = true;

l1.Visible = true;

pictureBox1.Dispose();

}

}

if (mode)

{

radius = float.Parse(sizeStarsNumber.Text);

radius2 = (float)(radius / 2.5);

float widthDraw;

float heighDraw;

if (maxValueC > 1)

{

colW = (int)((this.Width-40) / (2\*radius));

colH = (int)((this.Height -100)/ (2\*radius+30));

heighDraw = (float)(this.Height - 1 \* radius-45);

widthDraw = (float)(0 + 1\*radius);

}

else

{

colW = 1;

colH = 1;

widthDraw = (float)(this.Width / 2);

heighDraw = (float)(this.Height / 2);

}

int i = 0;

try

{

//это в класс

star.DrawStarGi(var1, var2, var3, var4, widthDraw, heighDraw, colorStars, ref G, i,maxValueC, checkMax, colW);

}

catch (Exception ez)

{

MessageBox.Show("Введите верное значение");

pictureBox1.Visible = false;

draw.Text = "НАРИСОВАТЬ";

typeStars.Visible = true;

sizeStars.Visible = true;

groupBox1.Visible = true;

sizeStarsNumber.Visible = true;

variants.Visible = true;

colorStars.Visible = true;

button4.Visible = true;

button1.Visible = true;

button3.Visible = true;

h1.Visible = true;

r1.Visible = true;

le1.Visible = true;

l1.Visible = true;

button1.Visible = true;

pictureBox1.Visible = false;

label1.Visible = false;

maxValue.Visible = false;

}

}

}

private PointF[] Calculate5StarPoints(PointF Orig, float outerradius, float innerradius)

{

double Ang36 = Math.PI / 5.0; // 36° x PI/180

double Ang72 = 2.0 \* Ang36; // 72° x PI/180 в лайфхакере так

float Sin36 = (float)Math.Sin(Ang36);

float Sin72 = (float)Math.Sin(Ang72);

float Cos36 = (float)Math.Cos(Ang36);

float Cos72 = (float)Math.Cos(Ang72);

PointF[] pnts = { Orig, Orig, Orig, Orig, Orig, Orig, Orig, Orig, Orig, Orig };

pnts[0].Y -= outerradius; // top off the star, or on a clock this is 12:00 or 0:00 hours

pnts[1].X += innerradius \* Sin36; pnts[1].Y -= innerradius \* Cos36; // 0:06 hours

pnts[2].X += outerradius \* Sin72; pnts[2].Y -= outerradius \* Cos72; // 0:12 hours

pnts[3].X += innerradius \* Sin72; pnts[3].Y += innerradius \* Cos72; // 0:18

pnts[4].X += outerradius \* Sin36; pnts[4].Y += outerradius \* Cos36; // 0:24

pnts[5].Y += innerradius;

pnts[6].X += pnts[6].X - pnts[4].X; pnts[6].Y = pnts[4].Y; // mirror point

pnts[7].X += pnts[7].X - pnts[3].X; pnts[7].Y = pnts[3].Y; // mirror point

pnts[8].X += pnts[8].X - pnts[2].X; pnts[8].Y = pnts[2].Y; // mirror point

pnts[9].X += pnts[9].X - pnts[1].X; pnts[9].Y = pnts[1].Y; // mirror point

return pnts;

}

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

{

}

private void radioButton1\_CheckedChanged(object sender, EventArgs e)

{

}

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

{

if (colorDialog1.ShowDialog() != DialogResult.Cancel)

{

colorStars.BackColor = colorDialog1.Color;

}

}

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

{

//Это в класс

//System.Diagnostics.Process.Start(@"c:\Temp\Downloads\some.doc");

System.Diagnostics.Process.Start(@"C:\Users\akime\OneDrive\Рабочий стол\4 КУРС МОЙ\varStars.docx");

}

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

{

}

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

{

if (tred2.IsAlive)

{

tred2.Resume();

button1.Enabled = false;

button3.Enabled = true;

button4.Enabled = true;

}

else

{

int z = 0;

if (double.Parse(sizeStarsNumber.Text) <= 500 && double.Parse(sizeStarsNumber.Text) >= 115)

{

//direction в класс и это

float direction = 0f;

if (r1.Checked)

{

direction = 0f;

}

if (l1.Checked)

{

direction = -360f;

}

if (h1.Checked)

{

direction = 360f;

}

if (le1.Checked)

{

direction = 0f;

z = 1;

}

button1.Enabled = false;

button3.Enabled = true;

button4.Enabled = true;

ClassLibrary1.Class1 s = new ClassLibrary1.Class1();

string forThred = sizeStarsNumber.Text + "|" + direction + '|' + z.ToString();

tred2 = new Thread(s.show);

tred2.Start(forThred);

//s.show(float.Parse(sizeStarsNumber.Text));

}

else

{

MessageBox.Show("Макимальный размер 500. Минимальный для Сириуса 115");

}

}

}

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

{

string namez = AppDomain.CurrentDomain.BaseDirectory + "MyWordDoc2.docx"; //это в класс

System.Diagnostics.Process.Start(namez);

}

private void button3\_Click\_1(object sender, EventArgs e)

{

tred2.Suspend(); //остановит, если остановит то ничего

button1.Enabled = true;

button3.Enabled = false;

button4.Enabled = false;

}

private void button4\_Click(object sender, EventArgs e)

{

tred2.Abort();//вызовет ошибку которая остановит поток

button3.Enabled = false;

button1.Enabled = true;

button3.Enabled = false;

}

private void toolStripButton1\_Click(object sender, EventArgs e)

{

}

private void toolStrip1\_ItemClicked(object sender, ToolStripItemClickedEventArgs e)

{

}

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

{

MessageBox.Show("Программа Егора Акимова. Рисование звёзд. \n \tСейчас " + DateTime.Now);

}

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

{

try

{

radius = float.Parse(sizeStarsNumber.Text); //два радиуса в класс

radius2 = (float)(radius / 2.5);

if ((Width - 15) / (2 \* radius) \* ((Height - 140) / (2 \* radius)) < Double.Parse(maxValue.Text))

{

throw new Exception("Не поместятся");

}

mode = !mode;

if (mode)

{

pictureBox1 = new PictureBox();

drawToolStripMenuItem.Text = "СМЕНИТЬ ЗВЕЗДУ";

typeStars.Visible = false;

sizeStars.Visible = false;

groupBox1.Visible = false;

sizeStarsNumber.Visible = false;

variants.Visible = false;

colorStars.Visible = false;

button1.Visible = false;

label1.Visible = false;

maxValue.Visible = false;

button4.Visible = false;

button1.Visible = false;

button3.Visible = false;

h1.Visible = false;

r1.Visible = false;

le1.Visible = false;

l1.Visible = false;

pictureBox1.Dock = DockStyle.Fill;

pictureBox1.BackColor = Color.White;

// Connect the Paint event of the PictureBox to the event handler method.

pictureBox1.Visible = true;

int z = 0;

pictureBox1.Paint += new PaintEventHandler(MyPainting);

this.Controls.Add(pictureBox1);

//в класс чтобы принт в ворд

wordApplicationForRun = new MyWord.Application(); // объект чтобы открыть ворд

string name = AppDomain.CurrentDomain.BaseDirectory; //это в класс как базу

MyWord.Document wordDoc = wordApplicationForRun.Documents.Add();

Range docRange = wordDoc.Range();

docRange.Text = "Курсовая Егор Акимов\n\n\n";

string imageName = name + "res.jpeg";

string namez = AppDomain.CurrentDomain.BaseDirectory + "MyWordDoc2.docx";

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

pictureBox1.DrawToBitmap(bmp, pictureBox1.ClientRectangle);

bmp.Save(name + "res.jpeg");

//pictureBox1.Image.Save(name + "MyWordDoc.png");

InlineShape pictureShape = docRange.InlineShapes.AddPicture(imageName);

wordDoc.SaveAs2(namez);

wordApplicationForRun.Quit();

this.SuspendLayout();

//this.ClientSize = new System.Drawing.Size(1220, 580);

//this.Name = "DrawingForm";

//this.Load += new System.EventHandler(this.DrawingForm\_Load);

this.ResumeLayout(false);

}

else

{

drawToolStripMenuItem.Text = "НАРИСОВАТЬ";

typeStars.Visible = true;

sizeStars.Visible = true;

sizeStarsNumber.Visible = true;

variants.Visible = true;

colorStars.Visible = true;

button1.Visible = true;

label1.Visible = true;

maxValue.Visible = true;

groupBox1.Visible = true;

button4.Visible = true;

button1.Visible = true;

button3.Visible = true;

h1.Visible = true;

r1.Visible = true;

le1.Visible = true;

l1.Visible = true;

pictureBox1.Dispose();

}

}

catch (Exception fe)

{

MessageBox.Show(fe.Message);

}

}

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

{

string namez = AppDomain.CurrentDomain.BaseDirectory + "MyWordDoc2.docx";

System.Diagnostics.Process.Start(namez);

}

private void button8\_Click(object sender, EventArgs e)

{

r1.Checked = true;

}

private void button5\_Click(object sender, EventArgs e)

{

h1.Checked = true;

}

private void button6\_Click(object sender, EventArgs e)

{

le1.Checked = true;

}

private void button7\_Click(object sender, EventArgs e)

{

l1.Checked = true;

}

private void aboutToolStripMenuItem1\_Click(object sender, EventArgs e)

{

MessageBox.Show("Программа Егора Акимова. Рисование звёзд. \n \tСейчас " + DateTime.Now);

}

private void toolStripSplitButton3\_ButtonClick(object sender, EventArgs e)

{

MessageBox.Show("Программа Егора Акимова. Рисование звёзд. \n \tСейчас " + DateTime.Now);

}

private void toolStripSplitButton2\_ButtonClick(object sender, EventArgs e)

{

try

{

radius = float.Parse(sizeStarsNumber.Text);

radius2 = (float)(radius / 2.5);

star.Radius1 = radius;

star.Radius2 = radius2;

if ((Width - 15) / (2 \* radius) \* ((Height - 140) / (2 \* radius)) < Double.Parse(maxValue.Text))

{

throw new Exception("Не поместятся");

}

mode = !mode;

if (mode)

{

pictureBox1 = new PictureBox();

drawToolStripMenuItem.Text = "СМЕНИТЬ ЗВЕЗДУ";

typeStars.Visible = false;

sizeStars.Visible = false;

groupBox1.Visible = false;

sizeStarsNumber.Visible = false;

variants.Visible = false;

colorStars.Visible = false;

button1.Visible = false;

label1.Visible = false;

maxValue.Visible = false;

button4.Visible = false;

button1.Visible = false;

button3.Visible = false;

h1.Visible = false;

r1.Visible = false;

le1.Visible = false;

l1.Visible = false;

pictureBox1.Dock = DockStyle.Fill;

pictureBox1.BackColor = Color.White;

// Connect the Paint event of the PictureBox to the event handler method.

pictureBox1.Visible = true;

int z = 0;

pictureBox1.Paint += new PaintEventHandler(MyPainting);

this.Controls.Add(pictureBox1);

//в класс из него взять

wordApplicationForRun = new MyWord.Application(); // объект чтобы открыть ворд

string name = AppDomain.CurrentDomain.BaseDirectory;

MyWord.Document wordDoc = wordApplicationForRun.Documents.Add();

Range docRange = wordDoc.Range();

docRange.Text = "Курсовая Егор Акимов\n\n\n";

string imageName = name + "res.jpeg";

string namez = AppDomain.CurrentDomain.BaseDirectory + "MyWordDoc2.docx";

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

pictureBox1.DrawToBitmap(bmp, pictureBox1.ClientRectangle);

bmp.Save(name + "res.jpeg");

//pictureBox1.Image.Save(name + "MyWordDoc.png");

InlineShape pictureShape = docRange.InlineShapes.AddPicture(imageName);

wordDoc.SaveAs2(namez);

wordApplicationForRun.Quit();

this.SuspendLayout();

//this.ClientSize = new System.Drawing.Size(1220, 580);

//this.Name = "DrawingForm";

//this.Load += new System.EventHandler(this.DrawingForm\_Load);

this.ResumeLayout(false);

}

else

{

drawToolStripMenuItem.Text = "НАРИСОВАТЬ";

typeStars.Visible = true;

sizeStars.Visible = true;

sizeStarsNumber.Visible = true;

variants.Visible = true;

colorStars.Visible = true;

button1.Visible = true;

label1.Visible = true;

maxValue.Visible = true;

groupBox1.Visible = true;

button4.Visible = true;

button1.Visible = true;

button3.Visible = true;

h1.Visible = true;

r1.Visible = true;

le1.Visible = true;

l1.Visible = true;

pictureBox1.Dispose();

}

}

catch (Exception fe)

{

MessageBox.Show(fe.Message);

}

}

private void helperToolStripMenuItem1\_Click(object sender, EventArgs e)

{

//в класс хелпер

//Help.ShowHelp(this, @"C:\Users\akime\OneDrive\Рабочий стол\4 КУРС МОЙ\РПВС ЛАБЫ\lab8\WindowsFormsApp1 — копия\WindowsFormsApp1\helph\NewProject.chm");

Help.ShowHelp(this, @"C:\Users\akime\OneDrive\Документы\GitHub\drawing-stars\curs2MB\res2mb\helper\MMyNewNewProject.chm");

}

private void toolStripButton1\_Click\_1(object sender, EventArgs e)

{

}

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

{

if (tred2.IsAlive)

{

tred2.Resume();

startItem.Enabled = false;

pauseItem.Enabled = true;

stopItem.Enabled = true;

}

else

{

int z = 0;

if (double.Parse(sizeStarsNumber.Text) <= 500 && double.Parse(sizeStarsNumber.Text) >= 115)

{

//direction в класс и это

float direction = 0f;

if (r1.Checked)

{

direction = 0f;

}

if (l1.Checked)

{

direction = -360f;

}

if (h1.Checked)

{

direction = 360f;

}

if (le1.Checked)

{

direction = 0f;

z = 1;

}

startItem.Enabled = false;

pauseItem.Enabled = true;

stopItem.Enabled = true;

ClassLibrary1.Class1 s = new ClassLibrary1.Class1();

string forThred = sizeStarsNumber.Text + "|" + direction + '|' + z.ToString();

tred2 = new Thread(s.show);

tred2.Start(forThred);

//s.show(float.Parse(sizeStarsNumber.Text));

}

else

{

MessageBox.Show("Макимальный размер 500. Минимальный для Сириуса 115");

}

}

}

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

{

tred2.Suspend(); //остановит, если остановит то ничего

startItem.Enabled = true;

pauseItem.Enabled = false;

stopItem.Enabled = false;

}

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

{

tred2.Abort();//вызовет ошибку которая остановит поток

pauseItem.Enabled = false;

startItem.Enabled = true;

pauseItem.Enabled = false;

}

private void toolStripSplitButton1\_ButtonClick(object sender, EventArgs e)

{

if (tred2.IsAlive)

{

tred2.Resume();

startItem.Enabled = false;

pauseItem.Enabled = true;

//stopItem.Enabled = true;

}

else

{

int z = 0;

if (double.Parse(sizeStarsNumber.Text) <= 500 && double.Parse(sizeStarsNumber.Text) >= 115)

{

//direction в класс и это

float direction = 0f;

if (r1.Checked)

{

direction = 0f;

}

if (l1.Checked)

{

direction = -360f;

}

if (h1.Checked)

{

direction = 360f;

}

if (le1.Checked)

{

direction = 0f;

z = 1;

}

button1.Enabled = false;

button3.Enabled = true;

button4.Enabled = true;

ClassLibrary1.Class1 s = new ClassLibrary1.Class1();

string forThred = sizeStarsNumber.Text + "|" + direction + '|' + z.ToString();

tred2 = new Thread(s.show);

tred2.Start(forThred);

//s.show(float.Parse(sizeStarsNumber.Text));

}

else

{

MessageBox.Show("Макимальный размер 500. Минимальный для Сириуса 115");

}

}

}

}

}

**Файл Program.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using Tao.FreeGlut;

using OpenGL;

namespace d33

{

class Program

{

private static int width = 1280, height = 720;

private static ShaderProgram program;

private static VBO<Vector3> star;

private static VBO<Vector3> starColor;

private static VBO<uint> starTriangles;

private static System.Diagnostics.Stopwatch watch;

private static float angle;

static float deltaTime = 0.001f;

public static void Main()

{

float max = 500;

float take = 315;

float res = take / max;

;

// create an OpenGL window

Glut.glutInit();

Glut.glutInitDisplayMode(Glut.GLUT\_DOUBLE | Glut.GLUT\_DEPTH);

Glut.glutInitWindowSize(width, height);

Glut.glutCreateWindow("OpenGL Tutorial");

// provide the Glut callbacks that are necessary for running this tutorial

Glut.glutIdleFunc(OnRenderFrame);

Glut.glutDisplayFunc(OnDisplay);

Glut.glutCloseFunc(OnClose);

// enable depth testing to ensure correct z-ordering of our fragments

Gl.Enable(EnableCap.DepthTest);

// compile the shader program

program = new ShaderProgram(VertexShader, FragmentShader);

// set the view and projection matrix, which are static throughout this tutorial

program.Use();

program["projection\_matrix"].SetValue(Matrix4.CreatePerspectiveFieldOfView(0.7f/res, (float)width / height, 0.1f, 1000f));

program["view\_matrix"].SetValue(Matrix4.LookAt(new Vector3(0, 0, 10), Vector3.Zero, new Vector3(0, 1, 0)));

double Ang36 = Math.PI / 5.0; // 36° x PI/180

double Ang72 = 2.0 \* Ang36; // 72° x PI/180 в лайфхакере так

float Sin36 = (float)Math.Sin(Ang36);

float Sin72 = (float)Math.Sin(Ang72);

float Cos36 = (float)Math.Cos(Ang36);

float Cos72 = (float)Math.Cos(Ang72);

// create a pyramid with vertices and colors

star = new VBO<Vector3>(new Vector3[] {

//new Vector3(0,-outerradius,0),

//new Vector3(innerrdius\*Sin36,-innerrdius\*Cos36, 0),

//new Vector3(outerradius\*Sin72,-outerradius\*Cos72,0),

//new Vector3(innerrdius\*Sin72,innerrdius\*Cos72+10, 0),

//new Vector3(outerradius\*Sin36,outerradius\*Cos36, 0),

//new Vector3(0,innerrdius, 0),

//new Vector3(-innerrdius\*Sin36,+innerrdius\*Cos36, 0),

//new Vector3(-outerradius\*Sin72,+outerradius\*Cos72,0),

//new Vector3(-innerrdius\*Sin72,+innerrdius\*Cos72, 0),

//new Vector3(-outerradius\*Sin36,-outerradius\*Cos36, 0),

new Vector3(0, 1, 0), new Vector3(-1, -1, 0), new Vector3(1, -1, 0), // front face

new Vector3(-1, -1, 0), new Vector3(-3, -2, 0), new Vector3(-1, -3, 0), // right face

new Vector3(-1,-3,0), new Vector3(0,-5,0),new Vector3(1,-3,0),

new Vector3(1,-3,0), new Vector3(3,-2,0), new Vector3(1,-1,0),

new Vector3(-1,-1,0), new Vector3(1,-1,0), new Vector3(1,-3,0),

new Vector3(-1,-1,0), new Vector3(-1,-3,0), new Vector3(1,-3,0)

//внутренность звезды

});

starColor = new VBO<Vector3>(new Vector3[] {

new Vector3(255, 237, 0), new Vector3(255, 237, 0), new Vector3(255, 237, 0),

new Vector3(255, 237, 0), new Vector3(255, 237, 0), new Vector3(255, 237, 0),

new Vector3(255, 237, 0), new Vector3(255, 237, 0), new Vector3(255, 237, 0),

new Vector3(255, 237, 0), new Vector3(255, 237, 0), new Vector3(255, 237, 0),

new Vector3(255, 237, 0), new Vector3(255, 237, 0), new Vector3(255, 237, 0),

new Vector3(255, 237, 0), new Vector3(255, 237, 0), new Vector3(255, 237, 0)});

starTriangles = new VBO<uint>(new uint[71] { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70 }, BufferTarget.ElementArrayBuffer);

watch = System.Diagnostics.Stopwatch.StartNew();

Glut.glutMainLoop();

}

private static void OnClose()

{

star.Dispose();

starColor.Dispose();

starTriangles.Dispose();

program.DisposeChildren = true;

program.Dispose();

}

private static void OnDisplay()

{

}

private static void OnRenderFrame()

{

watch.Stop();

watch.Restart();

angle += deltaTime;

Gl.Viewport(0, 0, width, height);

Gl.Clear(ClearBufferMask.ColorBufferBit | ClearBufferMask.DepthBufferBit);

Gl.UseProgram(program);

program["model\_matrix"].SetValue(Matrix4.CreateRotationY(angle) \* Matrix4.CreateTranslation(new Vector3(0.5f, 1.5, 0))); //отображение выше ниже

Gl.BindBufferToShaderAttribute(star, program, "vertexPosition");

Gl.BindBufferToShaderAttribute(starColor, program, "vertexColor");

Gl.BindBuffer(starTriangles);

Gl.DrawElements(BeginMode.Triangles, starTriangles.Count, DrawElementsType.UnsignedInt, IntPtr.Zero);

program["model\_matrix"].SetValue(Matrix4.CreateRotationY(angle / 2) \* Matrix4.CreateRotationX(angle) \* Matrix4.CreateTranslation(new Vector3(1.5f, 0, 0)));

Glut.glutSwapBuffers();

}

public static string VertexShader = @"

#version 130

in vec3 vertexPosition;

in vec3 vertexColor;

out vec3 color;

uniform mat4 projection\_matrix;

uniform mat4 view\_matrix;

uniform mat4 model\_matrix;

void main(void)

{

color = vertexColor;

gl\_Position = projection\_matrix \* view\_matrix \* model\_matrix \* vec4(vertexPosition, 1);

}

";

public static string FragmentShader = @"

#version 130

in vec3 color;

out vec4 fragment;

void main(void)

{

fragment = vec4(color, 1);

}

";

}

}

**Файл Star.cs**

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using MyWord = Microsoft.Office.Interop.Word;

namespace res2mb.Scripts

{

abstract class Star

{

private string wordDocName = AppDomain.CurrentDomain.BaseDirectory + "MyWordDoc2.docx";

private string baseDomain = AppDomain.CurrentDomain.BaseDirectory;

public string WordDocName { get => wordDocName; set => wordDocName = value; }

private float radius;

private float radius2;

private float direction;

public float Radius1 { get => radius; set => radius = value; }

public float Radius2 { get => radius2; set => radius2 = value; }

public float Direction { get => direction; set => direction = value; }

public abstract PointF[] Calculate5StarPoints(PointF Orig, float outerradius, float innerradius);

public abstract void DrawStarGi(RadioButton var1, RadioButton var2, RadioButton var3, RadioButton var4, float widthDraw, float heighDraw, Button colorStars, ref Graphics[] G, int i, int maxValueC, bool checkMax, int colW);

public abstract void ShowAllVariantsStars();

public abstract float chooseDirection(RadioButton r1, RadioButton l1, RadioButton h1, RadioButton le1);

public abstract void printInWord(MyWord.\_Application wordApplicationForRun, PictureBox pictureBox1);

}

}

**Файл StarImpl.cs**

using Microsoft.Office.Interop.Word;

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Drawing.Drawing2D;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using MyWord = Microsoft.Office.Interop.Word;

namespace res2mb.Scripts

{

class StarImpl : Star

{

public override PointF[] Calculate5StarPoints(PointF Orig, float outerradius, float innerradius)

{

double Ang36 = Math.PI / 5.0; // 36° x PI/180

double Ang72 = 2.0 \* Ang36; // 72° x PI/180 в лайфхакере так

float Sin36 = (float)Math.Sin(Ang36);

float Sin72 = (float)Math.Sin(Ang72);

float Cos36 = (float)Math.Cos(Ang36);

float Cos72 = (float)Math.Cos(Ang72);

PointF[] pnts = { Orig, Orig, Orig, Orig, Orig, Orig, Orig, Orig, Orig, Orig };

pnts[0].Y -= outerradius; //верх звезды 00 00

pnts[1].X += innerradius \* Sin36; pnts[1].Y -= innerradius \* Cos36; // 0:06 часов

pnts[2].X += outerradius \* Sin72; pnts[2].Y -= outerradius \* Cos72; // 0:12 часов

pnts[3].X += innerradius \* Sin72; pnts[3].Y += innerradius \* Cos72; // 0:18

pnts[4].X += outerradius \* Sin36; pnts[4].Y += outerradius \* Cos36; // 0:24

pnts[5].Y += innerradius;

pnts[6].X += pnts[6].X - pnts[4].X; pnts[6].Y = pnts[4].Y; // отражаем

pnts[7].X += pnts[7].X - pnts[3].X; pnts[7].Y = pnts[3].Y; // отражаем

pnts[8].X += pnts[8].X - pnts[2].X; pnts[8].Y = pnts[2].Y; // отражаем

pnts[9].X += pnts[9].X - pnts[1].X; pnts[9].Y = pnts[1].Y; // отражаем

return pnts;

}

public override void DrawStarGi(RadioButton var1, RadioButton var2, RadioButton var3, RadioButton var4, float widthDraw, float heighDraw, Button colorStars, ref Graphics[] G, int i, int maxValueC, bool checkMax, int colW)

{

while (checkMax && i < maxValueC)

{

if (var1.Checked)

{

PointF[] Star1 = Calculate5StarPoints(new PointF(widthDraw, heighDraw), Radius1, Radius2); //(x,y), длина внеешнего, длина внутреннего радиуса (50,20 топ)

SolidBrush FillBrush = new SolidBrush(colorStars.BackColor); //Внутренняя закраска

G[i].FillPolygon(FillBrush, Star1);

G[i].DrawPolygon(new Pen(Color.Purple, 5), Star1); //обводка

}

if (var2.Checked)

{

PointF[] Star2 = Calculate5StarPoints(new PointF(widthDraw, heighDraw), Radius1, Radius2);

HatchBrush pat = new HatchBrush(HatchStyle.Cross, Color.RosyBrown, colorStars.BackColor);

G[i].FillPolygon(pat, Star2);

}

if (var3.Checked)

{

PointF[] Star3 = Calculate5StarPoints(new PointF(widthDraw, heighDraw), Radius1, Radius2);

LinearGradientBrush lin = new LinearGradientBrush(new System.Drawing.Point(350, 100), new System.Drawing.Point(350, 500), colorStars.BackColor, Color.Cyan);

G[i].FillPolygon(lin, Star3);

}

if (var4.Checked)

{

PointF[] Star4 = Calculate5StarPoints(new PointF(widthDraw, heighDraw), Radius1, Radius2);

G[i].DrawPolygon(new Pen(colorStars.BackColor, 3), Star4);

}

if (maxValueC > 1) //смещам кажду звезду вправо

{

widthDraw += (float)(2 \*Radius1 + 0.05 \* Radius1);

}

i++;

if (maxValueC > 1)

{

if (i % colW == 0)

{

heighDraw -= (float)2.25 \* Radius1;

widthDraw = (float)(1.25 \* Radius1);

}

}

}

}

public override void ShowAllVariantsStars()

{

System.Diagnostics.Process.Start(@"C:\Users\akime\OneDrive\Рабочий стол\4 КУРС МОЙ\varStars.docx");

}

public override float chooseDirection(RadioButton r1, RadioButton l1, RadioButton h1, RadioButton le1)

{

if (r1.Checked)

{

Direction = 0f;

}

if (l1.Checked)

{

Direction = -360f;

}

if (h1.Checked)

{

Direction = 360f;

}

if (le1.Checked)

{

Direction = -0f;

}

return Direction;

}

public override void printInWord(MyWord.\_Application wordApplicationForRun, PictureBox pictureBox1)

{

wordApplicationForRun = new MyWord.Application(); // объект чтобы открыть ворд

string name = AppDomain.CurrentDomain.BaseDirectory; //это в класс как базу

MyWord.Document wordDoc = wordApplicationForRun.Documents.Add();

Range docRange = wordDoc.Range();

docRange.Text = "Курсовая Егор Акимов\n\n\n";

string imageName = name + "res.jpeg";

string namez = AppDomain.CurrentDomain.BaseDirectory + "MyWordDoc2.docx";

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

pictureBox1.DrawToBitmap(bmp, pictureBox1.ClientRectangle);

bmp.Save(name + "res.jpeg");

//pictureBox1.Image.Save(name + "MyWordDoc.png");

InlineShape pictureShape = docRange.InlineShapes.AddPicture(imageName);

wordDoc.SaveAs2(namez);

wordApplicationForRun.Quit();

}

}

}

**Файл Point.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace PendulumProject

{

public class Point

{

public virtual double X { get; set; }

public virtual double Y { get; set; }

}

}

**Файл Form1.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Drawing.Drawing2D;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace curs2MB

{

public partial class Form1 : Form

{

res2mb.Form2 formLogical;

public Form1()

{

InitializeComponent();

this.Text = "Курсовая Акимов Егор";

}

private void MyPainting(object sender, PaintEventArgs e)

{

Graphics G = e.Graphics;

G.SmoothingMode = SmoothingMode.HighQuality;

PointF[] Star1 = Calculate5StarPoints(new PointF(100f, 100f), 50f, 20f);

SolidBrush FillBrush = new SolidBrush(Color.Pink);

G.FillPolygon(FillBrush, Star1);

G.DrawPolygon(new Pen(Color.Purple, 5), Star1);

}

private PointF[] Calculate5StarPoints(PointF Orig, float outerradius, float innerradius)

{

double Ang36 = Math.PI / 5.0; // 36° x PI/180

double Ang72 = 2.0 \* Ang36; // 72° x PI/180 в лайфхакере так

float Sin36 = (float)Math.Sin(Ang36);

float Sin72 = (float)Math.Sin(Ang72);

float Cos36 = (float)Math.Cos(Ang36);

float Cos72 = (float)Math.Cos(Ang72);

PointF[] pnts = { Orig, Orig, Orig, Orig, Orig, Orig, Orig, Orig, Orig, Orig };

pnts[0].Y -= outerradius; // top off the star, or on a clock this is 12:00 or 0:00 hours

pnts[1].X += innerradius \* Sin36; pnts[1].Y -= innerradius \* Cos36; // 0:06 hours

pnts[2].X += outerradius \* Sin72; pnts[2].Y -= outerradius \* Cos72; // 0:12 hours

pnts[3].X += innerradius \* Sin72; pnts[3].Y += innerradius \* Cos72; // 0:18

pnts[4].X += outerradius \* Sin36; pnts[4].Y += outerradius \* Cos36; // 0:24

pnts[5].Y += innerradius;

pnts[6].X += pnts[6].X - pnts[4].X; pnts[6].Y = pnts[4].Y; // mirror point

pnts[7].X += pnts[7].X - pnts[3].X; pnts[7].Y = pnts[3].Y; // mirror point

pnts[8].X += pnts[8].X - pnts[2].X; pnts[8].Y = pnts[2].Y; // mirror point

pnts[9].X += pnts[9].X - pnts[1].X; pnts[9].Y = pnts[1].Y; // mirror point

return pnts;

}

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

{

}

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

{

}

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

{

//label1.Visible = false;

//label2.Visible = false;

//label3.Visible = false;

//button1.Visible = false;

//this.BackgroundImage = null;

//this.BackColor = Color.FromArgb(0, 0, 255);

formLogical = new res2mb.Form2();

this.Hide();

formLogical.Show();

}

}

}