**Московский авиационный институт**

**(Национальный исследовательский университет)**

Факультет №8 «Информационные технологии и прикладная математика»

Кафедра 806 «Вычислительной математики и программирования»

**Лабораторная работа №**

по курсу «Компьютерная графика»

Тема: Основы построения фотореалистичных изображений.

Студент: Сорокин Д.М.

Группа: 80-304Б

Преподаватель: Филлипов Г.С.

Оценка:

Москва, 2017

Задание:

Используя результаты Л.Р.№2, аппроксимировать заданное тело выпуклым многогранником. Точность аппроксимации задается пользователем. Обеспечить возможность вращения и масштабирования многогранника и удаление невидимых линий и поверхностей. Реализовать простую модель закраски для случая одного источника света. Параметры освещения и отражающие свойства материала задаются пользователем в диалоговом режиме.

Вариант:

15. Сектор эллипсоида.

Код:

Elipsoid.cs

**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** lab3CG

{

**class** Ellipsoid

{

**public** Ellipsoid(**int** NumberOfCircles, **int** NumberOfParabols)

{

triangleList = new List<Triangle>();

pointsList = new List<MyPoint>();

calcEllipsoid(NumberOfCircles, NumberOfParabols);

illuminate = **false**;

}

**public** **void** calcEllipsoid(**int** NumberOfCircles, **int** NumberOfParabols)

{

triangleList.Clear();

**double** t0 = 0, tn = 3, dt = (tn - t0) / NumberOfCircles;

List<MyPoint> points = new List<MyPoint>();

List<MyPoint> temp = new List<MyPoint>();

**for** (**int** i = 0; i < NumberOfCircles; ++i)

{

**double** x = t0 + i \* dt;

**double** z = Math.Sqrt(9 - x \* x);

points.**Add**(new MyPoint(x, 0, -z));

*//temp.Add(new MyPoint(x, 0, -z));*

}

*//points.AddRange(temp);*

**double** u0 = 0, un = 2 \* Math.PI, du = (un - u0) / NumberOfParabols;

**for** (**int** j = 0; j < NumberOfParabols; ++j)

{

List<MyPoint> cur\_points = new List<MyPoint>();

List<MyPoint> next\_points = new List<MyPoint>();

**double** angle = u0 + j \* du;

**double** next\_angle = u0 + (j + 1) \* du;

RotationMatrix r = new RotationMatrix('Z', angle);

RotationMatrix rr = new RotationMatrix('Z', next\_angle);

**for** (**int** i = 0; i < NumberOfCircles; ++i)

{

cur\_points.**Add**(r \* points[i]);

**if** (j == NumberOfParabols - 1)

{

next\_points.**Add**(points[i]);

pointsList.**Add**(points[i]);

}

**else**

{

next\_points.**Add**(rr \* points[i]);

pointsList.**Add**(rr \* points[i]);

}

}

**int** nn = cur\_points.Count;

**for** (**int** i = 0; i < nn - 1; ++i)

{

Triangle a = new Triangle(cur\_points[i], next\_points[i + 1], cur\_points[i + 1]);

Triangle b = new Triangle(cur\_points[i], next\_points[i], next\_points[i + 1]);

triangleList.**Add**(a);

triangleList.**Add**(b);

}

MyPoint center = new MyPoint(0, 0, 0);

Triangle top = new Triangle(cur\_points[nn - 1], next\_points[nn - 1], center);

triangleList.**Add**(top);

}

}

**public** **void** draw(Matrix preobr, Pen pen, Graphics g, MyPoint svet, **double**[,] iNofLight, **double**[,] kMat)

{

**for** (**int** i = 0; i < triangleList.Count; ++i)

{

Triangle t = preobr \* triangleList[i];

t.draw(pen, g, illuminate, svet, iNofLight, kMat);

}

}

**private** List<Triangle> triangleList;

**private** List<MyPoint> pointsList;

**public** **bool** illuminate;

}

}

Form1.cs

**private** **void** Form1\_Paint(**object** sender, PaintEventArgs e)

{

iNofLight[0, 0] = (**double**)buttonOfColor.BackColor.R \* ((**double**)trackBarRas.**Value** / 5) / 255;

iNofLight[0, 1] = (**double**)buttonOfColor.BackColor.G \* ((**double**)trackBarRas.**Value** / 5) / 255;

iNofLight[0, 2] = (**double**)buttonOfColor.BackColor.B \* ((**double**)trackBarRas.**Value** / 5) / 255;

iNofLight[1, 0] = (**double**)buttonOfColor.BackColor.R \* ((**double**)trackBarFon.**Value** / 5) / 255;

iNofLight[1, 1] = (**double**)buttonOfColor.BackColor.G \* ((**double**)trackBarFon.**Value** / 5) / 255;

iNofLight[1, 2] = (**double**)buttonOfColor.BackColor.B \* ((**double**)trackBarFon.**Value** / 5) / 255;

iNofLight[2, 0] = (**double**)buttonOfColor.BackColor.R \* ((**double**)trackBarMir.**Value** / 5) / 255;

iNofLight[2, 1] = (**double**)buttonOfColor.BackColor.G \* ((**double**)trackBarMir.**Value** / 5) / 255;

iNofLight[2, 2] = (**double**)buttonOfColor.BackColor.B \* ((**double**)trackBarMir.**Value** / 5) / 255;

kMat[0, 0] = **double**.Parse(textBoxRasR.Text) / 10;

kMat[0, 1] = **double**.Parse(textBoxRasG.Text) / 10;

kMat[0, 2] = **double**.Parse(textBoxRasB.Text) / 10;

kMat[1, 0] = **double**.Parse(textBoxFonR.Text) / 10;

kMat[1, 1] = **double**.Parse(textBoxFonG.Text) / 10;

kMat[1, 2] = **double**.Parse(textBoxFonB.Text) / 10;

kMat[2, 0] = **double**.Parse(textBoxMirR.Text) / 10;

kMat[2, 1] = **double**.Parse(textBoxMirG.Text) / 10;

kMat[2, 2] = **double**.Parse(textBoxMirB.Text) / 10;

Pen pen = new Pen(Color.Black, 1.0f);

e.Graphics.DrawRectangle(new Pen(Color.Red, 3.0f), (**int**)light.x, - (**int**)light.y, 1, 1);

**double** zoom\_level = (scale + mashtabK) / 1000;

**double** coeff = Math.Max(e.ClipRectangle.Width, e.ClipRectangle.Height) \* zoom\_level;

ShiftMatrix sh = new ShiftMatrix(e.ClipRectangle.Width / 2, e.ClipRectangle.Height / 2, 0);

ScalingMatrix sc = new ScalingMatrix(coeff, coeff, coeff);

RotationMatrix rtx = new RotationMatrix('X', my \* Math.PI / 180.0);

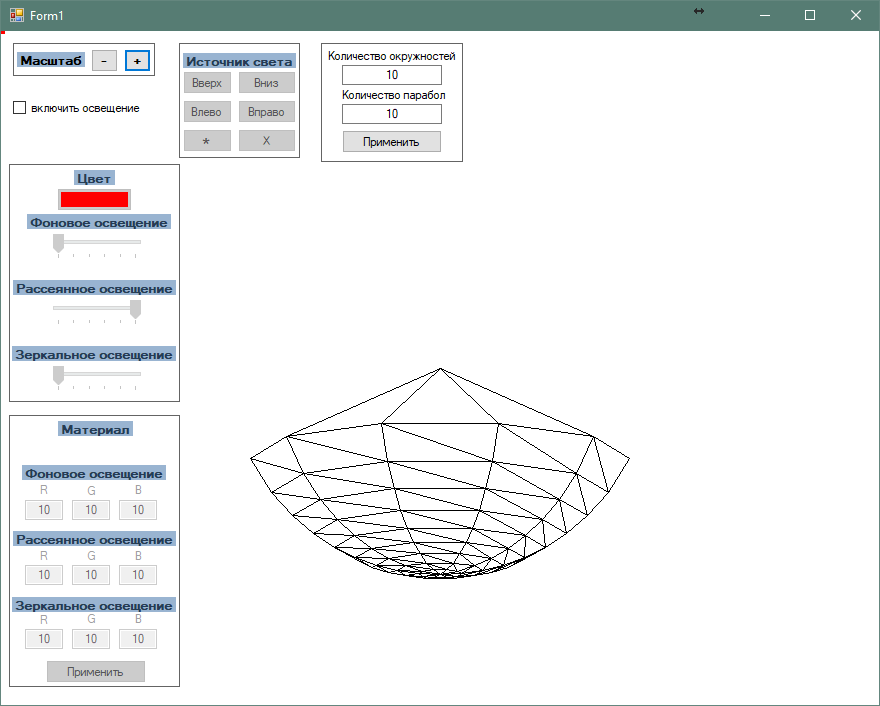
RotationMatrix rty = new RotationMatrix('Y', -mx \* Math.PI / 180.0);

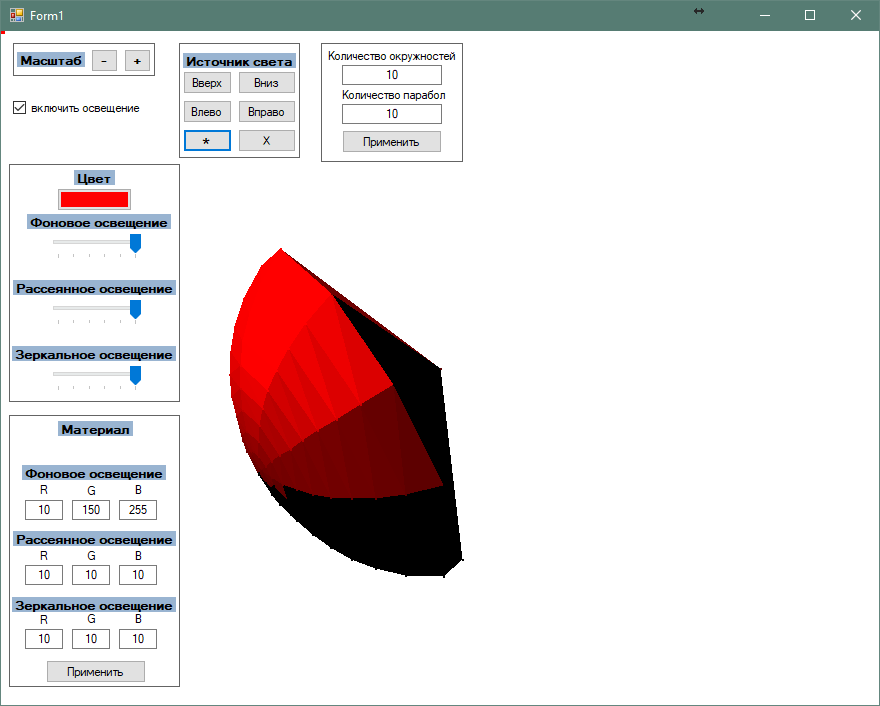
Matrix preobr = sh \* rtx \* rty \* sc;

ellipsoid.draw(preobr, pen, e.Graphics, light, iNofLight, kMat);

}

Примеры работы программы:





Выводы:

Я научился создавать фотореалистичные изображения. Узнал, как запрограммировать сложные 3D фигуры, а также их освещение и окраску. Данная лабораторная работа была очень увлекательна. Дальше интереснее!