МИНОБРНАУКИ РОССИИ  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Ижевский государственный технический университет имени М. Т. Калашникова»  
Институт «Информатика и вычислительная техника»  
Кафедра «Программное обеспечение»

Работа защищена с оценкой  
«\_\_\_\_\_\_\_\_\_\_»  
Дата \_\_\_\_\_\_\_\_\_\_\_\_  
Подпись\_\_\_\_\_\_\_\_\_/\_\_\_\_\_\_\_\_\_\_

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА  
к курсовой работе  
по дисциплине «Компьютерная графика»  
на тему: «Визуализация 3D-модели с использованием однородных координат»

Выполнил  
студент гр. Б18-191-2 Р. А. Гумметов

Руководитель  
к.т.н, доцент Л. Н. Левицкая

Рецензия:  
степень достижения поставленной цели работы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
полнота разработки темы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
уровень самостоятельности работы обучающегося\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
недостатки работы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc59405949)

[1. ПОСТАНОВКА ЗАДАЧИ 4](#_Toc59405950)

[2. ОПИСАНИЕ ПРОГРАММЫ 5](#_Toc59405951)

[2.1 Считывание координат из \*.obj файла 5](#_Toc59405952)

[2.2 Вращение модели 6](#_Toc59405953)

[2.3 Цвет модели 6](#_Toc59405954)

[2.4 Точка входа 6](#_Toc59405955)

[3. ПРИМЕР РАБОТЫ ПРОГРАММЫ 7](#_Toc59405956)

[ЗАКЛЮЧЕНИЕ 9](#_Toc59405957)

[СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ 10](#_Toc59405958)

[ПРИЛОЖЕНИЕ 11](#_Toc59405959)

# ВВЕДЕНИЕ

Компьютерная графика –раздел информатики, занимающийся проблемами создания и обработки на компьютере графических изображений.

Без компьютерной графики невозможно представить себе не только компьютерный, но и обычный, вполне материальный мир. Визуализация данных находит применение в самых разных сферах человеческой деятельности. Например, назовем медицину (компьютерная томография), научные исследования (визуализация строения вещества, векторных полей и других данных), моделирование тканей и одежды, опытно-конструкторские разработки.

Отдельным предметом считается трехмерная (3D) графика, изучающая приемы и методы построения объемных моделей объектов в виртуальном пространстве. Как правило, в ней сочетаются векторный и растровый способы формирования изображений.   
 Особенности цветового охвата характеризуют такие понятия, как черно-белая и цветная графика. На специализацию в отдельных областях указывают названия некоторых разделов: инженерная графика, научная графика, Web-графика, компьютерная полиграфия и прочие.

# 1. ПОСТАНОВКА ЗАДАЧИ

Требуется разработать программу, визуализирующую 3D-модель в формате \*.obj, относительно положения наблюдателя и направления его взгляда. Программа должна уметь считывать и анализировать \*.objфайлы, производить матричные преобразования и выводить объект на экран.

В программе должны быть заданы следующие матрицы:

1. матрица порта вывода;
2. матрица ортогонального проецирования;
3. матрица для перехода в систему координат наблюдателя;
4. матрица поворота.

Программа не должна использовать сторонние библиотеки. Все вычисления должны проводиться самостоятельно, а вывод на экран – осуществляться с помощью встроенных функций рисования простых фигур (линий, полигонов).

# 2. ОПИСАНИЕ ПРОГРАММЫ

Программа предназначена для вывода 3D изображения, описанного в \*.objфайле.Программа написана на языке C# с использованием WindowsFormsв среде разработки VisualStudio 2019.

2.1 Считывание координат из \*.obj файла

\*.objфайл содержит следующие данные:

1. позицию каждой вершины;
2. связь координат текстуры с вершиной;
3. нормаль для каждой вершины;
4. параметры, которые задают полигоны;
5. комментарии.

Вершина задаётся в строках, начинающихся с литеры v. Далее за ней располагаются координаты (x,y,z,w). Значение w по умолчанию равно 1.0.

Строка, начинающаяся с f, представляет собой полигон. Каждая поверхность (полигон) может состоять из трех или более вершин. Первое число после каждого пробела, это номер вершины в массиве вершин. Индексация начинается с первого элемента и может быть отрицательной – отрицательный индекс указывает позицию относительно последнего элемента.

Метод считывания и анализа \*.objфайла представлен в классе Sample. PictureRead() получает данные из файла, находит все вершины объекта и для каждой создает экземпляр класса Vector.

После определения всех вершин метод считывает данные о поверхности. Для каждой поверхности создается экземпляр класса Polygon.

## 2.2 Вращение модели

При нажатии на правую стрелку, объект начинает крутиться в правую сторону, при нажатии на левую стрелку - влево. При этом меняется угол обзора на модель, происходит умножение на матрицу поворота, которая по новому отрисовывает модель.

## 2.3 Цвет модели

С помощью класса Random, при каждом запуске выбираются оттенки красного, зеленого и синего цветов для модели. Поэтому при каждом запуске программы модель имеет новый цвет.

## 2.4 Точка входа

Программа начинает работу в методе Form1\_Load(). Этот метод вызывает функцию считывания из файла, определяет матрицы для преобразований и добавляет обработчики событий рисования и нажатия на клавишу. В зависимости от того, какая клавиша была нажата, определяется новый индекс в массиве, состоящем из заранее вычисленных поверхностей, и вызывается метод обновления экрана, после чего выводится новое изображение.

# 3. ПРИМЕР РАБОТЫ ПРОГРАММЫ

При запуске программы получаем изображение, представленное на рис. 1:

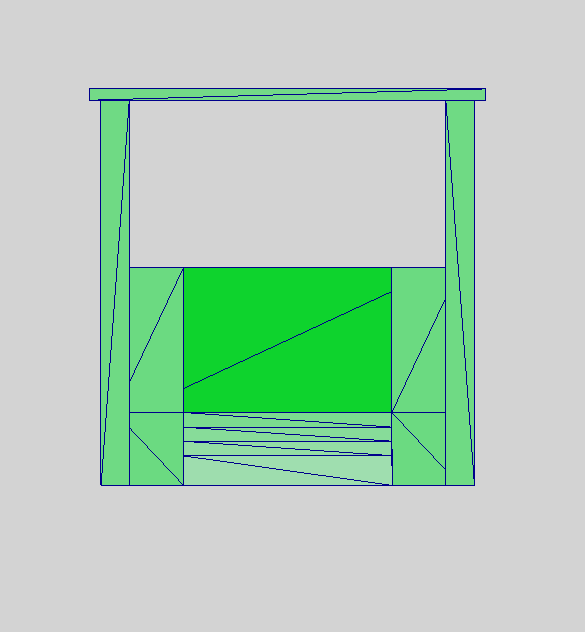


Рис. 1 - Изображение при запуске

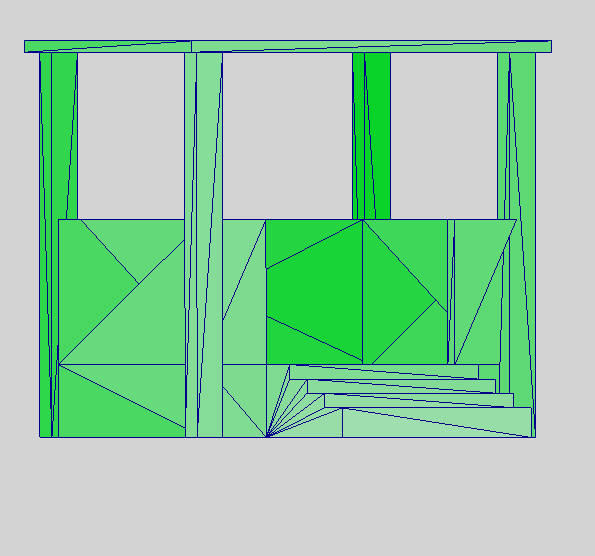


Рис. 2 - Изображение при нажатии на стрелку вправо

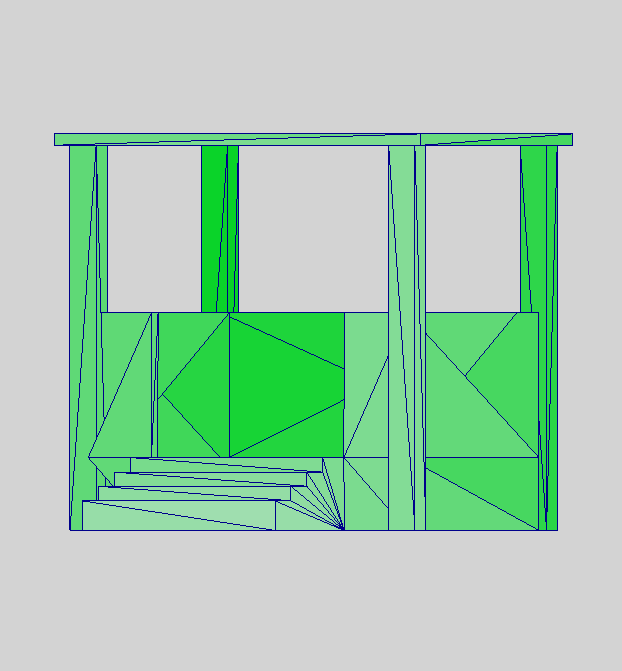


Рис. 3 - Изображение при нажатии на стрелку влево

# ЗАКЛЮЧЕНИЕ

В ходе курсовой работы разработана программа, визуализирующая 3Dмодель, описанную в \*.objфайле, относительно положения наблюдателя и направления его взгляда. Программа позволяет поворачивать объект вокруг оси Oy при нажатии на стрелки клавиатуры.

# СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1) Программирование с использованием OPENGL [электронный курс]. URL: https://www.opengl.org.ru (18.12.2020).

2) Введение в windows forms [электронный курс]. URL: https://metanit.com (18.12.2020).

3) Что такое 3D-графика и как она устроена [электронный курс]. URL: https://skillbox.ru (19.12.2020).

# ПРИЛОЖЕНИЕ

КОД ПРОГРАММЫ

Form1.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 MyWork

{

public partial class Form1 : Form

{

public int width, height;

Matrix ViewPortMatrix, MT;

double angle = 0;

const double dif = 2.5;

double delta = 0;

Color color;

private int r1;

private int r2;

private int g1;

private int g2;

private int b1;

private int b2;

private void Form1\_KeyDown(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Left)

{

delta = dif;

Invalidate();

}

else if (e.KeyCode == Keys.Right)

{

delta = -dif;

Invalidate();

}

}

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

{

if (angle < 0)

{

angle = 360 - Math.Abs(angle) % 360;

}

else

{

angle %= 360;

}

angle += delta;

Bitmap bitmap = new Bitmap(width, height);

Graphics g;

g = Graphics.FromImage(bitmap);

g.Clear(Color.LightGray);

Matrix TransRotMatrix = new Matrix();

TransRotMatrix \*= Matrix.RotateY(angle) \* Matrix.RotateY(180);

TFModel(MT \* TransRotMatrix, ref displaypoints);

double max = int.MinValue, min = int.MaxValue;

foreach (var pol in polygons)

{

if (pol.zAverage > max)

max = pol.zAverage;

if (pol.zAverage < min)

min = pol.zAverage;

}

var diff = max - min;

foreach (var pol in polygons)

{

var colint = (int)(((pol.zAverage - min) / (diff)) \* (255) + 0);

var rc = (int)(((pol.zAverage - min) / (diff)) \* (255 - r1) + r2);

var gc = (int)(((pol.zAverage - min) / (diff)) \* (255 - g1) + g2);

var bc = (int)(((pol.zAverage - min) / (diff)) \* (255 - b1) + b2);

g.FillPolygon(new SolidBrush(Color.FromArgb(rc,gc,bc)), pol.ToXYarray(ref displaypoints));

g.DrawPolygon(new Pen(Color.DarkBlue, 0.01f), pol.ToXYarray(ref displaypoints));

}

bitmap.RotateFlip(RotateFlipType.RotateNoneFlipY);

pictureBox1.Image = bitmap;

pictureBox1.Refresh();

}

public Form1()

{

InitializeComponent();

}

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

{

height = pictureBox1.Height;

width = pictureBox1.Width;

Random r = new Random();

color = Color.FromArgb(r.Next(0, 255), r.Next(0, 255), r.Next(0, 255));

r1 = color.R;

r2 = r.Next(0, r1);

g1 = color.G;

g2 = r.Next(0, g1);

b1 = color.B;

b2 = r.Next(0, b1);

PictureRead();

ViewPortMatrix = Matrix.Viewport(0, 0, width, height);

var coefficient = 2.2;

Matrix MP = Matrix.ProjectionOrthographic(points.Min(v => v.x) \* coefficient \* 2.2, points.Max(v => v.x) \* coefficient \* 2.2,

points.Min(v => v.y) \* coefficient - 1, points.Max(v => v.y) \* coefficient,

points.Min(v => v.z) \* coefficient, points.Max(v => v.z) \* coefficient);

MT = ViewPortMatrix \* MP;

var vectorEye = new Vector() { x = 0, y = 0, z = 1 };

var vectorCenter = new Vector() { x = 0, y = 0, z = 0 };

var vectorUp = new Vector() { x = 0, y = 1, z = 0 };

var ModelView = Matrix.LookAt(vectorEye, vectorCenter, vectorUp);

MT = MT \* ModelView;

timer1.Start();

}

}

}

Sample.cs

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Globalization;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace MyWork

{

public partial class Form1 : Form

{

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

List<Polygon> polygons = new List<Polygon>();

List<Vector> displaypoints = new List<Vector>();

public Polygon zAverageUpdate(Polygon polygon)

{

double sum = 0.0;

for (int i = 0; i < polygon.vertex.Count; i++)

{

sum += displaypoints[polygon.vertex[i]].z;

}

polygon.zAverage = sum / polygon.vertex.Count;

return polygon;

}

public struct Polygon

{

public List<int> vertex;

public double zAverage;

public PointF[] ToXYarray(ref List<Vector> pointslist)

{

var list = new List<PointF>();

foreach (var point in vertex)

{

list.Add(pointslist[point].ToXY());

}

return list.ToArray();

}

}

public void PolsSort()

{

polygons.Sort(delegate (Polygon a, Polygon b) {

return a.zAverage.CompareTo(b.zAverage);

});

}

public void PictureRead()

{

string pathToPicture = "guard post.obj";

StreamReader fileobj = new StreamReader(pathToPicture, Encoding.UTF8);

points = new List<Vector>();

string str;

while (!fileobj.EndOfStream)

{

str = fileobj.ReadLine();

if (str.StartsWith("v") && !str.StartsWith("vn") && !str.StartsWith("vt"))

{

str = str.TrimEnd();

var pointsl = str.Split(' ').Skip(1).Select(v => Double.Parse(v.Replace('.', ','))).ToArray();

points.Add(new Vector(pointsl[0], pointsl[1], pointsl[2]));

}

else if (str.StartsWith("f"))

{

str = str.TrimEnd();

var pols = str.Split(' ').Skip(1).Select(v => int.Parse(v.Split('/').First())).ToArray();

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

{

//if (pols[i] > 0)

pols[i] = pols[i] - 1;

//else

//pols[i] = points.Count + pols[i];

}

Polygon polygon = new Polygon();

polygon.vertex = pols.ToList();

if (polygon.vertex.Count != 0)

polygons.Add(polygon);

}

}

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

foreach (var polygon in polygons)

{

Polygon t = polygon;

for (int i = 0; i < polygon.vertex.Count; i++)

{

t.zAverage += points[polygon.vertex[i]].z;

}

t.zAverage /= polygon.vertex.Count;

temp.Add(t);

}

polygons = temp;

fileobj.Close();

}

public void TFModel(Matrix m, ref List<Vector> listpoints)

{

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

foreach (var point in points)

{

temp.Add(point.MatrixMultVector(m));

}

listpoints = temp;

List<Polygon> t = new List<Polygon>();

foreach (var pol in polygons)

{

t.Add(zAverageUpdate(pol));

}

polygons = t;

PolsSort();

}

}

}

Matrices.cs

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Linq;

using System.Text;

namespace MyWork

{

public class Vector

{

public double x, y, z, w;

public PointF ToXY()

{

return new PointF((float)x, (float)y);

}

public Vector()

{

x = 0.0;

y = 0.0;

z = 0.0;

w = 1.0;

}

public Vector(double x, double y, double z)

{

this.x = x;

this.y = y;

this.z = z;

w = 1.0;

}

public Vector(double x, double y, double z, double w)

{

this.x = x;

this.y = y;

this.z = z;

this.w = w;

}

public Vector MatrixMultVector(Matrix m) //Умножение матрицы на вектор

{

double[] vector = new double[4];

vector[0] = x;

vector[1] = y;

vector[2] = z;

vector[3] = w;

double[] arr\_res = new double[4];

for (int i = 0; i < 4; i++)

{

for (int j = 0; j < 4; j++)

{

arr\_res[i] += m.uMatrix[i, j] \* vector[j];

}

}

Vector v\_res = new Vector(arr\_res[0], arr\_res[1], arr\_res[2], arr\_res[3]);

return v\_res;

}

static public Vector operator ^(Vector v1, Vector v2) //Умножение двух векторов

{

Vector v\_res = new Vector(v1.y \* v2.z - v1.z \* v2.y, v1.z \* v2.x - v1.x \* v2.z, v1.x \* v2.y - v1.y \* v2.x);

return v\_res;

}

static public Vector operator -(Vector v1, Vector v2) //Вычитание векторов

{

Vector v\_res = new Vector(v1.x - v2.x, v1.y - v2.y, v1.z - v2.z);

return v\_res;

}

public double LenVec() //длина вектора

{

double lenVec = Math.Sqrt(x \* x + y \* y + z \* z);

return lenVec;

}

public void Normalize() //нормализация вектора

{

x /= LenVec();

y /= LenVec();

z /= LenVec();

}

}

public class Matrix

{

public double[,] uMatrix;

public Matrix()

{

uMatrix = new double[4, 4];

for (int i = 0; i < 4; i++)

uMatrix[i, i] = 1.0;

}

static public Matrix operator \*(Matrix m1, Matrix m2) //Умножение матрицы на матрицу

{

double[,] arr = new double[4, 4];

for (int i = 0; i < 4; i++)

{

for (int j = 0; j < 4; j++)

{

for (int k = 0; k < 4; k++)

{

arr[i, j] += m1.uMatrix[i, k] \* m2.uMatrix[k, j];

}

}

}

Matrix m\_res = new Matrix();

for (int i = 0; i < 4; i++)

{

for (int j = 0; j < 4; j++)

{

m\_res.uMatrix[i, j] = arr[i, j];

}

}

return m\_res;

}

static public Matrix Viewport(int x, int y, int w, int h) //Матрица окна вывода, меняет мои координаты на координаты экрана

{

Matrix m\_res = new Matrix();

m\_res.uMatrix[0, 3] = x + w / 2.0;

m\_res.uMatrix[1, 3] = y + h / 2.0;

m\_res.uMatrix[2, 3] = 255.0 / 2.0;

m\_res.uMatrix[0, 0] = w / 2.0;

m\_res.uMatrix[1, 1] = h / -2.0;

m\_res.uMatrix[2, 2] = 255.0 / 2.0;

return m\_res;

}

static public Matrix ProjectionOrthographic(double l, double r, double t, double b, double n, double f)

{

var matrix = new Matrix();

matrix.uMatrix[0, 0] = 2.0 / (r - l);

matrix.uMatrix[1, 1] = 2.0 / (t - b);

matrix.uMatrix[2, 2] = -2.0 / (f - n);

matrix.uMatrix[0, 3] = -(r + l) / (r - l);

matrix.uMatrix[1, 3] = -(t + b) / (t - b);

matrix.uMatrix[2, 3] = -(f + n) / (f - n);

matrix.uMatrix[3, 3] = 1;

return matrix;

}

public static Matrix LookAt(Vector eye, Vector center, Vector up)

{

var matrix = new Matrix();

var Z = eye - center;

Z.Normalize();

var X = up ^ Z;

X.Normalize();

var Y = Z ^ X;

Y.Normalize();

matrix.uMatrix[0, 0] = X.x;

matrix.uMatrix[0, 1] = X.y;

matrix.uMatrix[0, 2] = X.z;

matrix.uMatrix[0, 3] = -center.x;

matrix.uMatrix[1, 0] = Y.x;

matrix.uMatrix[1, 1] = Y.y;

matrix.uMatrix[1, 2] = Y.z;

matrix.uMatrix[1, 3] = -center.y;

matrix.uMatrix[2, 0] = Z.x;

matrix.uMatrix[2, 1] = Z.y;

matrix.uMatrix[2, 2] = Z.z;

matrix.uMatrix[2, 3] = -center.z;

matrix.uMatrix[3, 3] = 1;

return matrix;

}

static public Matrix Scale(double sx, double sy, double sz)

{

Matrix m\_res = new Matrix();

m\_res.uMatrix[0, 0] = sx;

m\_res.uMatrix[1, 1] = sy;

m\_res.uMatrix[2, 2] = sz;

return m\_res;

}

static public Matrix Translated(double tx, double ty, double tz)

{

Matrix m\_res = new Matrix();

m\_res.uMatrix[0, 3] = tx;

m\_res.uMatrix[1, 3] = ty;

m\_res.uMatrix[2, 3] = tz;

return m\_res;

}

static public Matrix Rotate(double x\_alfa,double y\_alfa,double z\_alfa)

{

double rad = 0.0174533;

double alfa = rad \* x\_alfa + rad \* y\_alfa + rad \* z\_alfa;

Matrix m\_res = new Matrix();

if (z\_alfa != 0)

{

m\_res.uMatrix[0, 0] = Math.Cos(alfa);

m\_res.uMatrix[1, 0] = Math.Sin(alfa);

m\_res.uMatrix[0, 1] = -Math.Sin(alfa);

m\_res.uMatrix[1, 1] = Math.Cos(alfa);

}

if (x\_alfa != 0)

{

m\_res.uMatrix[1, 1] = Math.Cos(alfa);

m\_res.uMatrix[1, 2] = -Math.Sin(alfa);

m\_res.uMatrix[2, 1] = Math.Sin(alfa);

m\_res.uMatrix[2, 2] = Math.Cos(alfa);

}

if (y\_alfa != 0)

{

m\_res.uMatrix[0, 0] = Math.Cos(alfa);

m\_res.uMatrix[0, 2] = Math.Sin(alfa);

m\_res.uMatrix[2, 0] = -Math.Sin(alfa);

m\_res.uMatrix[2, 2] = Math.Cos(alfa);

}

return m\_res;

}

static public Matrix RotateX(double x\_alfa)

{

double alfa = x\_alfa \* Math.PI / 180.0;

Matrix m\_res = new Matrix();

if (x\_alfa != 0)

{

m\_res.uMatrix[0, 0] = 1;

m\_res.uMatrix[1, 1] = Math.Cos(alfa);

m\_res.uMatrix[1, 2] = -Math.Sin(alfa);

m\_res.uMatrix[2, 1] = Math.Sin(alfa);

m\_res.uMatrix[2, 2] = Math.Cos(alfa);

m\_res.uMatrix[3, 3] = 1;

}

return m\_res;

}

static public Matrix RotateY(double y\_alfa)

{

double alfa = y\_alfa \* Math.PI / 180.0;

Matrix m\_res = new Matrix();

if (y\_alfa != 0)

{

m\_res.uMatrix[0, 0] = Math.Cos(alfa);

m\_res.uMatrix[0, 2] = Math.Sin(alfa);

m\_res.uMatrix[1, 1] = 1;

m\_res.uMatrix[2, 0] = -Math.Sin(alfa);

m\_res.uMatrix[2, 2] = Math.Cos(alfa);

}

return m\_res;

}

static public Matrix RotateZ(double z\_alfa)

{

double alfa = z\_alfa \* Math.PI / 180.0;

Matrix m\_res = new Matrix();

if (z\_alfa != 0)

{

m\_res.uMatrix[0, 0] = Math.Cos(alfa);

m\_res.uMatrix[1, 0] = Math.Sin(alfa);

m\_res.uMatrix[0, 1] = -Math.Sin(alfa);

m\_res.uMatrix[1, 1] = Math.Cos(alfa);

m\_res.uMatrix[2, 2] = 1;

}

return m\_res;

}

}

}