**Московский авиационный институт**

**(Национальный исследовательский университет)**

Факультет №8 «Информационные технологии и прикладная математика»

Кафедра 806 «Вычислительной математики и программирования»

**Лабораторная работа № 2**

по курсу «Компьютерная графика»

Тема: Каркасная визуализация выпуклого многогранника. Удаление невидимых линий.

Студент: Сорокин Д.М.

Группа: 80-304Б

Преподаватель: Филлипов Г.С.

Оценка:

Москва, 2017

Задание:

Разработать формат представления многогранника и процедуру его каркасной отрисовки в ортографической и изометрической проекциях. Обеспечить удаление невидимых линий и возможность пространственных поворотов и масштабирования многогранника. Обеспечить автоматическое центрирование и изменение размеров изображения при изменении размеров окна.

Вариант:

15. Гранная прямая правильная пирамида

Код:

Matrix.cs – класс матрицы

**using** System;

**namespace** lab2CG

{

**class** Matrix

{

**public** Matrix()

{

m = new **double**[4,4];

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

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

m[i,j] = 0;

}

*// умножение матриц*

**public** **static** Matrix **operator** \*(Matrix left, Matrix right)

{

Matrix res = new Matrix();

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

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

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

res.m[i,j] += left.m[i,k] \* right.m[k,j];

**return** res;

}

*// умножение матриц на число*

**public** **static** Matrix **operator** \*(Matrix left, **double** digit)

{

Matrix res = new Matrix();

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

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

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

res.m[i, j] = left.m[i, j] \* digit;

**return** res;

}

*// умножение вектора на матрицу*

**public** **static** MyPoint **operator** \*(Matrix left, MyPoint p)

{

**double**[] t = {0, 0, 0, 0};

**double**[] tp = {p.x, p.y, p.z, p.w};

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

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

t[i] += left.m[i,j] \* tp[j];

**return** new MyPoint(t[0], t[1], t[2], t[3]);

}

*// присваивание*

**public** **static** Matrix **operator** ==(Matrix left, Matrix right)

{

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

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

left.m[i,j] = right.m[i,j];

**return** left;

}

**public** **static** Matrix **operator** !=(Matrix left, Matrix right)

{

**return** left;

}

**public** **double**[,] m;

};

*//матрица масштабирования*

**class** ScalingMatrix : Matrix

{

**public** ScalingMatrix(**double** a, **double** b, **double** c)

{

m[0, 0] = a;

m[1, 1] = b;

m[2, 2] = c;

m[3, 3] = 1;

}

}

*//матрица поворота*

**class** RotationMatrix: Matrix

{

**public** RotationMatrix(**char** axis, **double** a)

{

**switch**(axis)

{

**case** 'X':

m[1,1] = m[2,2] = Math.Cos(a);

m[1,2] = -1 \*(m[2,1] = Math.Sin(a));

m[0,0] = 1;

**break**;

**case** 'Y':

m[0,0] = m[2,2] = Math.Cos(a);

m[2,0] = -1 \*(m[0,2] = Math.Sin(a));

m[1,1] = 1;

**break**;

**case** 'Z':

m[0,0] = m[1,1] = Math.Cos(a);

m[1,0] = -1 \*(m[0,1] = Math.Sin(a));

m[2,2] = 1;

**break**;

}

m[3,3] = 1;

}

}

*//матрица сдвига*

**class** ShiftMatrix: Matrix

{

**public** ShiftMatrix(**double** a, **double** b, **double** c)

{

m[0,3] = a;

m[1,3] = b;

m[2,3] = c;

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

m[i,i] = 1;

}

}

}

MyPoint.cs – класс вектора

**namespace** lab2CG

{

**class** MyPoint

{

**public** MyPoint(**double** x = 0, **double** y = 0, **double** z = 0, **double** w = 1)

{

**this**.x = x;

**this**.y = y;

**this**.z = z;

**this**.w = w;

}

**public** MyPoint(MyPoint rhs)

{

x = rhs.x;

y = rhs.y;

z = rhs.z;

w = rhs.w;

}

*// операции над векторами*

*// вычитание*

**public** **static** MyPoint **operator** -(MyPoint left, MyPoint right)

{

**return** new MyPoint(left.x - right.x, left.y - right.y, left.z - right.z, left.w - right.w);

}

*// присваивание*

**public** **static** MyPoint **operator** ==(MyPoint thisPoint, MyPoint rhs)

{

thisPoint.x = rhs.x;

thisPoint.y = rhs.y;

thisPoint.z = rhs.z;

thisPoint.w = rhs.w;

**return** thisPoint;

}

**static** **public** MyPoint **operator** !=(MyPoint thisPoint, MyPoint rhs)

{

**return** thisPoint;

}

*// скалярное произведение*

**static** **public** **double** **operator** \*(MyPoint left, MyPoint right)

{

**return** left.x \* right.x + left.y \* right.y + left.z \* right.z;

}

*// векторное произведение*

**static** **public** MyPoint **operator** ^(MyPoint left, MyPoint right)

{

MyPoint res = new MyPoint();

res.x = left.y \* right.z - left.z \* right.y;

res.y = left.z \* right.x - left.x \* right.z;

res.z = left.x \* right.y - left.y \* right.x;

res.w = 0;

**return** res;

}

**public** **double** x, y, z, w;

}

}

Triangle.cs – класс треугольника

**using** System.Drawing;

**namespace** lab2CG

{

**class** Triangle

{

**public** Triangle(MyPoint a, MyPoint b, MyPoint c)

{

**this**.a = a;

**this**.b = b;

**this**.c = c;

}

**public** MyPoint n()

{

**return** (b - a) ^ (c - a);

}

**public** **static** Triangle **operator** \*(Matrix m, Triangle t)

{

**return** new Triangle(m \* t.a, m \* t.b, m \* t.c);

}

**public** **void** Draw(Pen pen, Graphics g, **bool** isWithInvLin)

{

**if** (!isWithInvLin)

{

**if** (n().z > 0)

{

g.DrawLine(pen, (**int**)a.x, (**int**)a.y, (**int**)b.x, (**int**)b.y);

g.DrawLine(pen, (**int**)b.x, (**int**)b.y, (**int**)c.x, (**int**)c.y);

g.DrawLine(pen, (**int**)c.x, (**int**)c.y, (**int**)a.x, (**int**)a.y);

}

}

**else**

{

g.DrawLine(pen, (**int**)a.x, (**int**)a.y, (**int**)b.x, (**int**)b.y);

g.DrawLine(pen, (**int**)b.x, (**int**)b.y, (**int**)c.x, (**int**)c.y);

g.DrawLine(pen, (**int**)c.x, (**int**)c.y, (**int**)a.x, (**int**)a.y);

}

}

**public** MyPoint a, b, c;

}

}

Основные функции из form1.cs

**public** **void** calcPyramid()

{

**double** radius = **double**.Parse(textBoxOfPirRadius.Text);

**double** hight = **double**.Parse(textBoxOfPirHight.Text);

List<Triangle> pyramid = new List<Triangle>();

**double** katet = radius / Math.Sqrt(2);

MyPoint A = new MyPoint(0, 0, -hight);

MyPoint B = new MyPoint(radius, 0, -hight);

MyPoint C = new MyPoint(0, radius, -hight);

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

MyPoint E = new MyPoint(0, -radius, -hight);

MyPoint F = new MyPoint(-radius, 0, -hight);

MyPoint M = new MyPoint(katet, katet, -hight);

MyPoint N = new MyPoint(-katet, katet, -hight);

MyPoint K = new MyPoint(-katet, -katet, -hight);

MyPoint H = new MyPoint(katet, -katet, -hight);

Triangle a = new Triangle(B, M, A);

Triangle b = new Triangle(M, C, A);

Triangle c = new Triangle(C, N, A);

Triangle d = new Triangle(N, F, A);

Triangle e = new Triangle(F, K, A);

Triangle f = new Triangle(A, K, E);

Triangle g = new Triangle(A, E, H);

Triangle h = new Triangle(A, H, B);

Triangle j = new Triangle(D, M, B);

Triangle k = new Triangle(D, C, M);

Triangle l1 = new Triangle(D, N, C);

Triangle m = new Triangle(D, F, N);

Triangle n = new Triangle(D, K, F);

Triangle l = new Triangle(D, E, K);

Triangle o = new Triangle(D, H, E);

Triangle p = new Triangle(D, B, H);

pyramid.**Add**(a);

pyramid.**Add**(b);

pyramid.**Add**(c);

pyramid.**Add**(d);

pyramid.**Add**(e);

pyramid.**Add**(f);

pyramid.**Add**(g);

pyramid.**Add**(h);

pyramid.**Add**(j);

pyramid.**Add**(k);

pyramid.**Add**(l);

pyramid.**Add**(m);

pyramid.**Add**(n);

pyramid.**Add**(l1);

pyramid.**Add**(o);

pyramid.**Add**(p);

surfaces.**Add**(pyramid);

}

List<List<Triangle>> surfaces;

*// текущие координаты курсора и координаты его предыдущего положения*

**int** mx, my, cx, cy;

*// индекс текущей поверхности и масштаб*

**int** cur;

**float** scale;

**double** mashtabK;

**bool** isMouseDown;

**private** **void** mashtabMinusButton\_Click(**object** sender, EventArgs e)

{

mashtabK -= 5;

Refresh();

}

**private** **void** mashtabPlusButton\_Click(**object** sender, EventArgs e)

{

mashtabK += 5;

Refresh();

}

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

{

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

**double** zoomLevel = (scale + mashtabK) / 1000;

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

Matrix m1 = new Matrix();

m1.m[0, 0] = Math.Sqrt(3);

m1.m[0, 1] = 0;

m1.m[0, 2] = -Math.Sqrt(3);

m1.m[1, 0] = 1;

m1.m[1, 1] = 2;

m1.m[1, 2] = 1;

m1.m[2, 0] = Math.Sqrt(2);

m1.m[2, 1] = -Math.Sqrt(2);

m1.m[2, 2] = Math.Sqrt(2);

m1.m[3, 3] = 1;

m1 = m1 \* (1 / Math.Sqrt(6));

Matrix m2 = new Matrix();

m2.m[0, 0] = 1;

m2.m[0, 1] = 0;

m2.m[0, 2] = 0;

m2.m[1, 0] = 0;

m2.m[1, 1] = 1;

m2.m[1, 2] = 0;

m2.m[2, 0] = 0;

m2.m[2, 1] = 0;

m2.m[2, 2] = 0;

m2.m[3, 3] = 1;

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 tr = new Matrix();

**if** (izometrCheckBox.Checked) { tr = sc \* m1 \* m2 \* rtx \* rty; tr = (sh \* 2) \* tr; }

**else** tr = sh \* sc \* rtx \* rty;

**for** (**int** i = 0; i < surfaces[cur].Count; ++i)

{

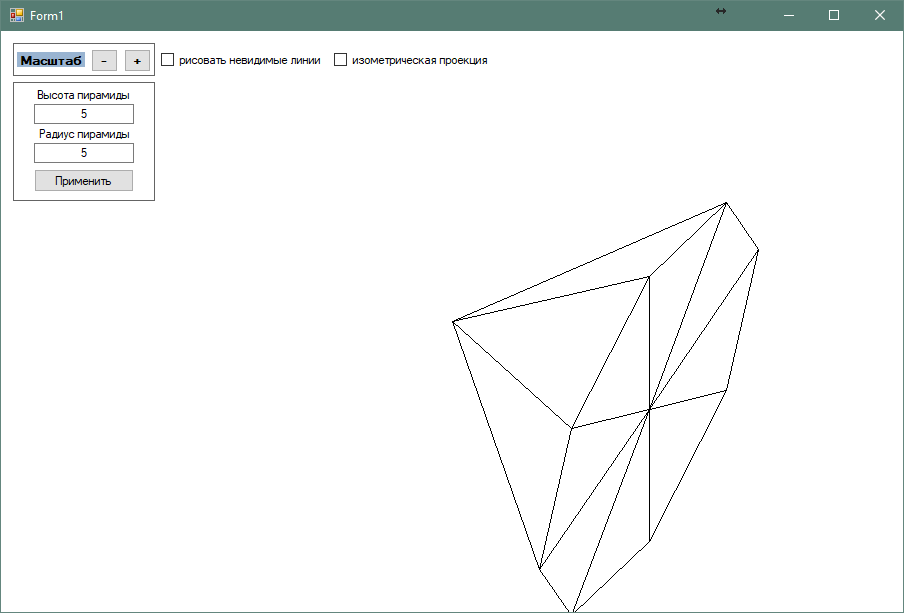
Triangle t = tr \* surfaces[cur][i];

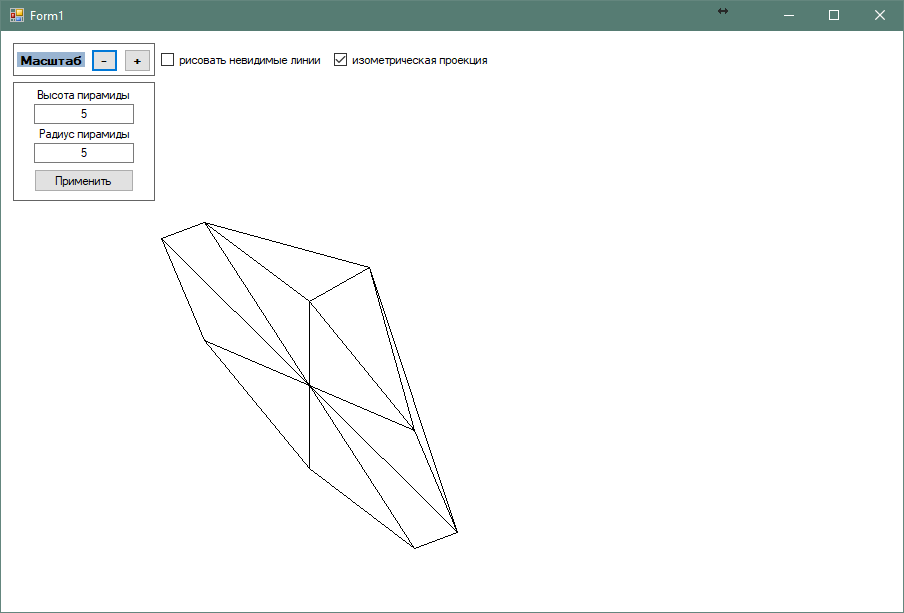
t.Draw(pen, e.Graphics, invLinCheckBox.Checked);

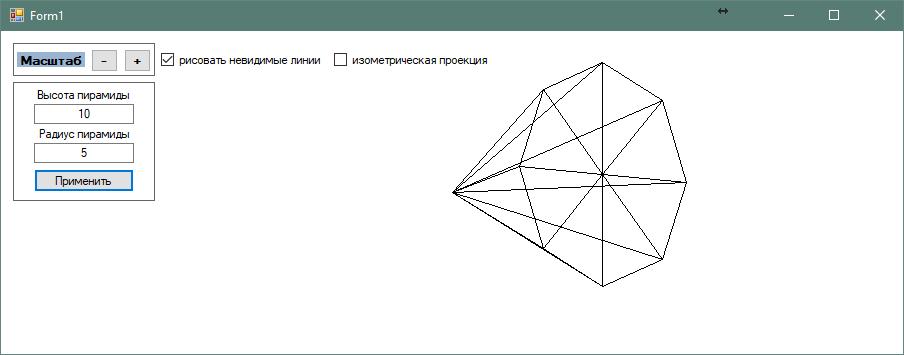
}

}

Примеры работы программы:







Выводы:

Выполнив данную лабораторную работу, я приобрел навыки визуализации каркасных выпуклых многогранных фигур на компьютере. Наработки, созданные в этой лабораторной работе, пригодятся мне в последующих.