**Московский авиационный институт**

**(Национальный исследовательский университет)**

Институт: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

Дисциплина: «Компьютерная графика»

**Лабораторная работа № 2**

Тема: Каркасная визуализация выпуклого многогранника. Удаление невидимых линий.

Студент: Шиляева Н. С.

Группа: 80-304

Преподаватель: Чернышов Л.Н.

Дата:

Оценка:

Москва, 2020

1. Постановка задачи

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 16**:** 16-гранная правильная пирамида.

1. Решение задачи

Используемый язык программирования: C#.

Полигональная сетка представляет собой поверхность многогранника аппроксимирующего тело заданным способом. В зависимости от способа аппроксимации многогранник может находиться внутри аппроксимируемого тела, пересекать его или заключать в себе. В последнем случае, если многогранник выпуклый, полигональная сетка является выпуклой оболочкой тела. Грани многогранника, как правило, представляют собой выпуклые многоугольники, зачастую треугольники, что связано с удобством реализации для треугольников ряда алгоритмов (закраска, вычисление пересечений граней и т.д.). Построение аппроксимирующего многогранника с треугольными гранями называется триангуляцией. Вершины полигональной сетки удобно расположить на поверхности тела по некоторому закону. Очевидно, что при выбранном законе расположения вершин, точность аппроксимации данного тела будет определяться величиной разбиения. Каркасное изображение тела представляет собой совокупность линий, расположенных на поверхности тела или вблизи нее.

При выборе способа аппроксимации, криволинейную поверхность, заданную в виде F(x,y,z) = 0, часто бывает удобно представить в эквивалентной параметрической форме x = x(u,v),

y = y(u,v),

z = z(u,v), u0<= u<= up , v0<=v<=vp .

При фиксированных значениях параметра v = v i = v 0 +i\*v , i=0,1,…,N v , где N v +1 –количество кривых постоянного значения параметра v, аv = (vp - v0 )/N v - шаг по параметру v, получаем семейство кривых x = x(u, v i ), y = y(u, v i ), z = z(u, v i ). Аналогично при фиксированных значениях параметра u получаем семейство кривых x = x(u j ,v), y = y(u j ,v) ,z= z(u j ,v). Все эти кривые лежат на аппроксимируемой поверхности и точки их пересечения, соответствующие парам (u j ,v i) , можно выбрать в качестве вершин полигональной сетки.

1. Набор тестов

Вводим параметры пирамиды, изменяем масштаб, также можно посмотреть невидимые линии и изометрическую проекцию.

Тест 1:



Тест 2:



Тест 3:



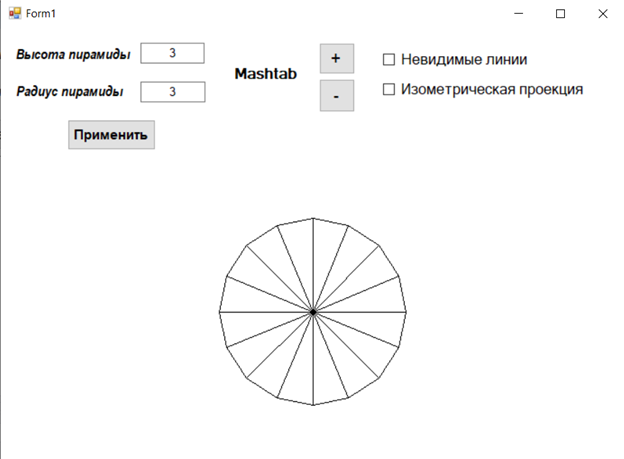
Тест 4:



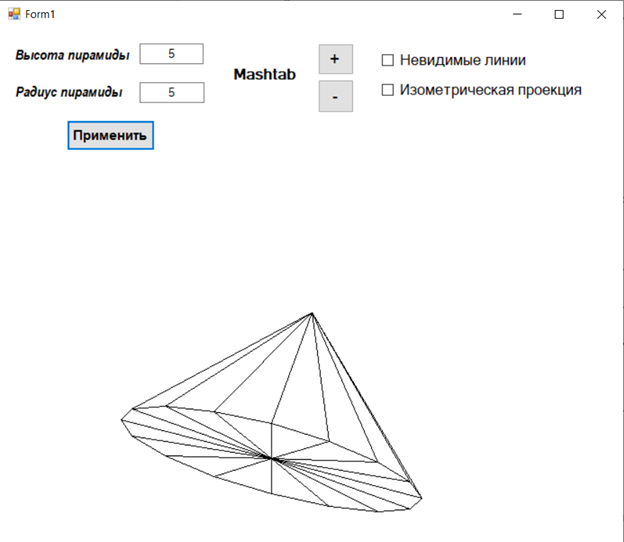
1. Результаты выполнения тестов

Наглядно видим выполнение программы по тестам, описанным ранее.

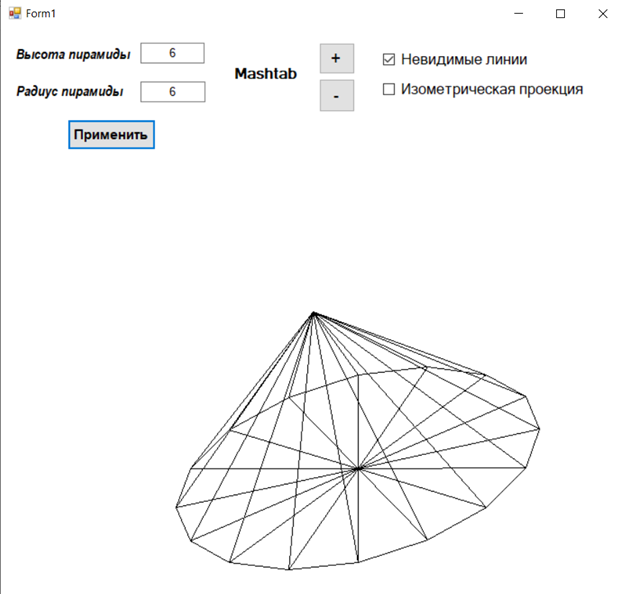
Тест 1:



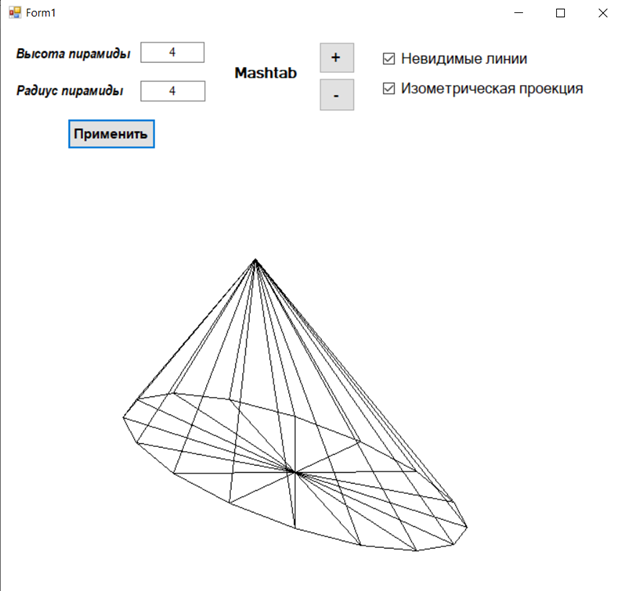
Тест 2:



Тест 3:



Тест 4:



1. Листинг программы

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 LR\_2

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

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

calcPyramid();

mx = 0; my = 0; cx = 0; cy = 0; cur = 0;

scale = 50;

mashtabK = 0;

isMouseDown = false;

}

public void calcPyramid()

{

double radius = double.Parse(textBox\_radius.Text);

double hight = double.Parse(textBox\_height.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);

MyPoint A1 = new MyPoint(-radius \* Math.Sin(Math.PI / 8), -radius \* Math.Cos(Math.PI / 8), -hight);

MyPoint A2 = new MyPoint(-radius \* Math.Cos(Math.PI / 8), -radius \* Math.Sin(Math.PI / 8), -hight);

MyPoint A3 = new MyPoint(-radius \* Math.Cos(Math.PI / 8), radius \* Math.Sin(Math.PI / 8), -hight);

MyPoint A4 = new MyPoint(-radius \* Math.Sin(Math.PI / 8), radius \* Math.Cos(Math.PI / 8), -hight);

MyPoint A21 = new MyPoint(radius \* Math.Sin(Math.PI / 8), -radius \* Math.Cos(Math.PI / 8), -hight);

MyPoint A22 = new MyPoint(radius \* Math.Cos(Math.PI / 8), -radius \* Math.Sin(Math.PI / 8), -hight);

MyPoint A23 = new MyPoint(radius \* Math.Cos(Math.PI / 8), radius \* Math.Sin(Math.PI / 8), -hight);

MyPoint A24 = new MyPoint(radius \* Math.Sin(Math.PI / 8), radius \* Math.Cos(Math.PI / 8), -hight);

Triangle t1 = new Triangle(E, A, A1);

Triangle t2 = new Triangle(A1, A, K);

Triangle t3 = new Triangle(K, A, A2);

Triangle t4 = new Triangle(A2, A, F);

Triangle t5 = new Triangle(F, A, A3);

Triangle t6 = new Triangle(A3, A, N);

Triangle t7 = new Triangle(N, A, A4);

Triangle t8 = new Triangle(A4, A, C);

Triangle t11 = new Triangle(E, A1, D);

Triangle t12 = new Triangle(A1, K, D);

Triangle t13 = new Triangle(K, A2, D);

Triangle t14 = new Triangle(A2, F, D);

Triangle t15 = new Triangle(F, A3, D);

Triangle t16 = new Triangle(A3, N, D);

Triangle t17 = new Triangle(N, A4, D);

Triangle t18 = new Triangle(A4, C, D);

Triangle t21 = new Triangle(A, E, A21);

Triangle t22 = new Triangle(A21, H, A);

Triangle t23 = new Triangle(H, A22, A);

Triangle t24 = new Triangle(A22, B, A);

Triangle t25 = new Triangle(B, A23, A);

Triangle t26 = new Triangle(A23, M, A);

Triangle t27 = new Triangle(M, A24, A);

Triangle t28 = new Triangle(A24, C, A);

Triangle t211 = new Triangle(E, D, A21);

Triangle t212 = new Triangle(A21, D, H);

Triangle t213 = new Triangle(H, D, A22);

Triangle t214 = new Triangle(A22, D, B);

Triangle t215 = new Triangle(B, D, A23);

Triangle t216 = new Triangle(A23, D, M);

Triangle t217 = new Triangle(M, D, A24);

Triangle t218 = new Triangle(A24, D, C);

pyramid.Add(t1);

pyramid.Add(t2);

pyramid.Add(t3);

pyramid.Add(t4);

pyramid.Add(t5);

pyramid.Add(t6);

pyramid.Add(t7);

pyramid.Add(t8);

pyramid.Add(t11);

pyramid.Add(t12);

pyramid.Add(t13);

pyramid.Add(t14);

pyramid.Add(t15);

pyramid.Add(t16);

pyramid.Add(t17);

pyramid.Add(t18);

pyramid.Add(t21);

pyramid.Add(t22);

pyramid.Add(t23);

pyramid.Add(t24);

pyramid.Add(t25);

pyramid.Add(t26);

pyramid.Add(t27);

pyramid.Add(t28);

pyramid.Add(t211);

pyramid.Add(t212);

pyramid.Add(t213);

pyramid.Add(t214);

pyramid.Add(t215);

pyramid.Add(t216);

pyramid.Add(t217);

pyramid.Add(t218);

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 Button\_Minus\_Click(object sender, EventArgs e)

{

mashtabK--;

this.Refresh();

}

private void Button\_Plus\_Click(object sender, EventArgs e)

{

mashtabK++;

this.Refresh();

}

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

{

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

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

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

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 (checkBox\_isometric\_view.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, checkBox1\_invisible\_lines.Checked);

}

}

private void Form1\_MouseMove(object sender, MouseEventArgs e)

{

if (isMouseDown)

{

int delta\_x = e.X - cx;

int delta\_y = e.Y - cy;

mx += delta\_x;

my += delta\_y;

cx = e.X;

cy = e.Y;

this.Refresh();

}

}

private void Form1\_MouseDown(object sender, MouseEventArgs e)

{

isMouseDown = true;

cx = e.X;

cy = e.Y;

}

private void Form1\_MouseUp(object sender, MouseEventArgs e)

{

isMouseDown = false;

}

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

{

this.Refresh();

}

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

{

}

private void CheckBox1\_invisible\_lines\_CheckedChanged(object sender, EventArgs e)

{

this.Refresh();

}

private void Button\_apply\_Click(object sender, EventArgs e)

{

surfaces.Clear();

calcPyramid();

this.Refresh();

}

private void CheckBox\_isometric\_view\_CheckedChanged(object sender, EventArgs e)

{

surfaces.Clear();

calcPyramid();

this.Refresh();

}

}

}

1. Вывод

Успешно разработан формат представления многогранника и процедуру его каркасной отрисовки в ортографической и изометрической проекциях. Обеспечено удаление невидимых линий и возможность пространственных поворотов и

масштабирования многогранника. Обеспечено автоматическое центрирование и изменение размеров изображения при изменении размеров окна.

Список литературы

1. Шилдт, Герберт. Ш57 С# 4.0: полное руководство. : Пер. с англ. — М. : ООО "И.Д. Вильямс", 2011. — 1056 с.: ил. — Парал. тит. англ. (дата обращения: 03.10.2020).
2. Прайс. C# 7 и .NET Core. Кросс-платформенная разработка для профессионалов, 3-е издание. -М.: Питер, 2018 - 640 с. (дата обращения: 04.10.2020).