МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

**Український державний університет науки і технологій**

Кафедра «Комп’ютерні інформаційні технології»

**Лабораторна робота №3**

**з дисципліни «Комп’ютерна графіка»**

**на тему:** **«Системи координат та афінні перетворення.»**

Виконав: студент гр. ПЗ2011

Кулик С. В.

Прийняла: Шаповал І. В.

.

Дніпро, 2022

**Тема.** Тривимірна графіка.

**Мета.** Вивчення методів формування 3D-зображень.

**Завдання**

Розробити програму для відображення каркасу двох заданих фігур та куска поверхні у трьохвимірному просторі.

Передбачити:

• афінні перетворення кожної фігури, поверхні;

• можливість для користувача задавати параметри перетворень;

• зображення фігури при заданому виді проектування (ізометрія чи диметрія);

• зображення проекцій фігур на площини XOY, XOZ, YOZ;

• зображення вісей з підписами;

• при зображенні фігур передбачити скриття невидимих ліній чи позначення їх пунктирними лініями.

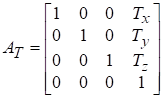
**Індивідуальне завдання**

Трикутна усічена піраміда. Диметрія. Форма Ерміта.

**Теоретичні відомості**

Рівняння переміщення  записуються у вигляді матриці перетворень в однорідних координатах:

,                                              (4.1)

де  – матриця переміщення.

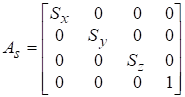
Рівняння масштабування відносно початку координат:

                                               (4.2)

де – масштабні множники.

У матричному вигляді рівняння масштабування записується як:

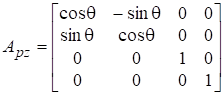
,                                               (4.3)

де  – матриця масштабування.

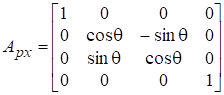
У матричному вигляді рівняння повороту записується як:

                                               (4.4)

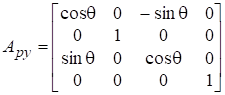
В тривимірному просторі поворот навколо осі  описує матриця повороту :

                                 (4.5)

У тривимірному просторі поворот навколо осі  описує матриця повороту :

                                 (4.6)

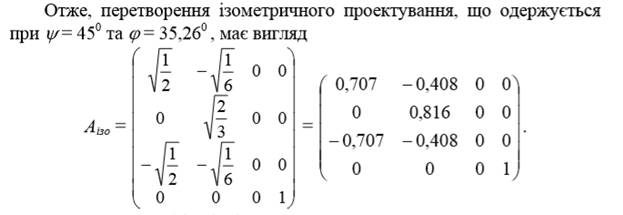
У тривимірному просторі поворот навколо осі  описує матриця повороту :

                                (4.7)

Якщо треба зробити одразу декілька перетворень, наприклад, два, то це можна зробити шляхом перемноження матриць  та  таким чином:

  тобто застосовується одна матриця, а потім множиться на , де .

**Ізометрична проекція** є видом аксонометричної проекції. Нормаль до площини проекції складає рівні кути з кожної із головних координатних осей. Ця проекція характеризується тим, що всі три головні координатні осі однаково скорочуються, тому можна проводити вимірювання впродовж напрямку осей з однаковим масштабом, а також вони проеціюються так, що їх проекції мають рівні кути (120) між друг другом.



Для опису параметричних кубічних кривих існують деякі методи: **метод Ерміта**, в якому задаються положення кінцевих точок кривої та дотичні вектори до них; **метод Без’є**, в якому задається положення кінцевих точок кривої, а для неявного завдання дотичних в цих точках використовуються ще дві інші точки, яки звичайно лежать не на кривій; **метод В-сплайнів**, де кінцеві точки не лежать на кривій, в результаті чого перша і друга похідні є неперервними на кінцях сегмента.

**Текст програми**

***Form1.cs***

using System;

using System.Windows.Forms;

namespace ComputerGraphic3

{

public partial class Form1 : Form

{

SurfaceErmit surface;

Figure figure;

Graph3d currentObject;

public Form1()

{

InitializeComponent();

listBoxSurface.SelectedIndex = 0;

listBoxParametr.SelectedIndex = 0;

figure = new Figure(3, 50, 150, 1, 200, new Point3d(200, 50, 50));

Point3d[] m = {

new Point3d(100,100,0), new Point3d(200, 100, 0), new Point3d(0, 0, 200),new Point3d(0, 0, 0),

new Point3d(100,300,0), new Point3d(200, 300, 0), new Point3d(0, 0, 0),new Point3d(0, 0, 0),

new Point3d(0, 0, 0), new Point3d(0, 0, 0), new Point3d(0, 0, 0),new Point3d(0, 0, 0),

new Point3d(0, 0, 0), new Point3d(0, 0, 0), new Point3d(0, 0, 0),new Point3d(0, 0, 0)

};

surface = new SurfaceErmit(m);

currentObject = figure;

}

private void pictureBox1\_Paint(object sender, PaintEventArgs e)

{

ProjType[] pr = { ProjType.front, ProjType.left, ProjType.top, ProjType.dimetry };

currentObject.draw(e.Graphics, pr[listBoxParametr.SelectedIndex]);

}

private void listBoxParametr\_SelectedIndexChanged(object sender, EventArgs e)

{

pictureBox1.Refresh();

}

private void listBoxSurface\_SelectedIndexChanged(object sender, EventArgs e)

{

if (listBoxSurface.SelectedIndex == 0)

currentObject = figure;

else

currentObject = surface;

pictureBox1.Refresh();

}

private void comboBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

int i = int.Parse(comboBox1.Text);

int j = int.Parse(comboBox2.Text);

Point3d p = surface.points[i \* 4 + j];

textBoxX.Text = p.x.ToString("0.##");

textBoxY.Text = p.y.ToString("0.##");

textBoxZ.Text = p.z.ToString("0.##");

pictureBox1.Refresh();

}

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

{

int i = int.Parse(comboBox1.Text);

int j = int.Parse(comboBox2.Text);

Point3d p = new Point3d(double.Parse(textBoxX.Text),

double.Parse(textBoxY.Text),

double.Parse(textBoxZ.Text));

surface.points[i \* 4 + j] = p;

pictureBox1.Refresh();

}

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

{

currentObject.move(

checkBoxX.Checked ? 10 : 0,

checkBoxY.Checked ? 10 : 0,

checkBoxZ.Checked ? 10 : 0

);

pictureBox1.Refresh();

}

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

{

currentObject.move(

checkBoxX.Checked ? -10 : 0,

checkBoxY.Checked ? -10 : 0,

checkBoxZ.Checked ? -10 : 0

);

pictureBox1.Refresh();

}

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

{

currentObject.scale(

checkBoxX.Checked ? 1.1 : 1,

checkBoxY.Checked ? 1.1 : 1,

checkBoxZ.Checked ? 1.1 : 1

);

pictureBox1.Refresh();

}

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

{

currentObject.scale(

checkBoxX.Checked ? 0.91919 : 1,

checkBoxY.Checked ? 0.91919 : 1,

checkBoxZ.Checked ? 0.91919 : 1

);

pictureBox1.Refresh();

}

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

{

currentObject.rotate(

checkBoxX.Checked ? 10 : 0,

checkBoxY.Checked ? 10 : 0,

checkBoxZ.Checked ? 10 : 0

);

pictureBox1.Refresh();

}

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

{

currentObject.rotate(

checkBoxX.Checked ? -10 : 0,

checkBoxY.Checked ? -10 : 0,

checkBoxZ.Checked ? -10 : 0

);

pictureBox1.Refresh();

}

}

}

***Graph3D.cs***

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Text;

namespace ComputerGraphic3

{

public enum ProjType { front, left, top, dimetry };

public struct Point3d

{

public double x, y, z;

public Point3d(double x = 0, double y = 0, double z = 0)

{

this.x = x;

this.y = y;

this.z = z;

}

public PointF project(ProjType projectType, bool surface)

{

if (!surface)

{

switch (projectType)

{

case ProjType.front:

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

case ProjType.left:

return new PointF((float)z + 15, (float)y);

case ProjType.top:

return new PointF((float)z + 20, (float)x - 100);

case ProjType.dimetry:

return new PointF(150 + (float)(0.935 \* x - 0.354 \* z), 125 + (float)(-0.118 \* x + 0.943 \* y - 0.312 \* z));

}

}

else

{

switch (projectType)

{

case ProjType.front:

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

case ProjType.left:

return new PointF(300 + (float)z, (float)y);

case ProjType.top:

return new PointF(300 + (float)z, (float)x - 100);

case ProjType.dimetry:

return new PointF(25 + (float)(0.935 \* x - 0.354 \* z), 50 + (float)(-0.118 \* x + 0.943 \* y - 0.312 \* z));

}

}

return new PointF(0, 0);

}

};

public interface Graph3d

{

void draw(Graphics g, ProjType pt);

void move(int dx, int dy, int dz);

void scale(double kx, double ky, double kz);

void rotate(double ax, double ay, double az);

}

public class Figure : Graph3d

{

List<Point3d> points;

int num, step;

bool surface;

//int num - point number

//int step - draw

//int topR, int botR - radiuses

//int h - height

//Point3d cnt - center

public Figure(int num, int topRadius, int botRadius, int step, int height, Point3d cnt)

{

points = new List<Point3d>();

this.num = num;

this.step = step;

surface = false;

//top

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

{

double x = cnt.x + topRadius \* Math.Cos(360.0 / num \* i \* Math.PI / 180.0);

double z = cnt.z + topRadius \* Math.Sin(360.0 / num \* i \* Math.PI / 180.0);

points.Add(new Point3d(x, cnt.y, z));

}

//bottom

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

{

double x = cnt.x + botRadius \* Math.Cos(360.0 / num \* i \* Math.PI / 180.0);

double z = cnt.z + botRadius \* Math.Sin(360.0 / num \* i \* Math.PI / 180.0);

points.Add(new Point3d(x, cnt.y + height, z));

}

}

public void draw(Graphics g, ProjType pt)

{

//top

List<PointF> edge = new List<PointF>();

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

{

edge.Add(points[i].project(pt, surface));

}

edge.Add(edge[0]);

g.DrawLines(Pens.Black, edge.ToArray());

//bottom

edge.Clear();

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

{

edge.Add(points[num + i].project(pt, surface));

}

edge.Add(edge[0]);

g.DrawLines(Pens.Black, edge.ToArray());

//side

for (int i = 0; i < num; i += step)

g.DrawLine(Pens.Black, points[i].project(pt, surface), points[i + num].project(pt, surface));

}

public void move(int dx, int dy, int dz)

{

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

{

Point3d p = points[i];

points[i] = new Point3d(

p.x + dx,

p.y + dy,

p.z + dz

);

}

}

Point3d center()

{

Point3d p = new Point3d(0, 0, 0);

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

{

p.x += points[i].x;

p.y += points[i].y;

p.z += points[i].z;

}

p.x /= points.Count;

p.y /= points.Count;

p.z /= points.Count;

return p;

}

public void rotate(double ax, double ay, double az)

{

var cnt = center();

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

{

if (ax != 0)

{

Point3d p = new Point3d(points[i].x - cnt.x, points[i].y - cnt.y, points[i].z - cnt.z);

points[i] = new Point3d(

cnt.x + p.x,

cnt.y + p.y \* Math.Cos(ax \* Math.PI / 180) - p.z \* Math.Sin(ax \* Math.PI / 180),

cnt.z + p.y \* Math.Sin(ax \* Math.PI / 180) + p.z \* Math.Cos(ax \* Math.PI / 180)

);

}

if (ay != 0)

{

Point3d p = new Point3d(points[i].x - cnt.x, points[i].y - cnt.y, points[i].z - cnt.z);

points[i] = new Point3d(

cnt.x + p.x \* Math.Cos(ay \* Math.PI / 180) - p.z \* Math.Sin(ay \* Math.PI / 180),

cnt.y + p.y,

cnt.z + p.x \* Math.Sin(ay \* Math.PI / 180) + p.z \* Math.Cos(ay \* Math.PI / 180)

);

}

if (az != 0)

{

Point3d p = new Point3d(points[i].x - cnt.x, points[i].y - cnt.y, points[i].z - cnt.z);

points[i] = new Point3d(

cnt.x + p.x \* Math.Cos(az \* Math.PI / 180) - p.y \* Math.Sin(az \* Math.PI / 180),

cnt.y + p.x \* Math.Sin(az \* Math.PI / 180) + p.y \* Math.Cos(az \* Math.PI / 180),

cnt.z + p.z

);

}

}

}

public void scale(double kx, double ky, double kz)

{

var cnt = center();

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

{

Point3d p = new Point3d(points[i].x - cnt.x, points[i].y - cnt.y, points[i].z - cnt.z);

points[i] = new Point3d(

cnt.x + p.x \* kx,

cnt.y + p.y \* ky,

cnt.z + p.z \* kz

);

}

}

}

public class Matrix4x4

{

int n, m;

public double[,] elem = new double[4, 4];

public Matrix4x4(int n, int m)

{

this.n = n;

this.m = m;

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

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

elem[i, j] = 0;

}

public static Matrix4x4 operator \*(Matrix4x4 m1, Matrix4x4 m2)

{

Matrix4x4 res = new Matrix4x4(m1.n, m2.m);

for (int i = 0; i < res.m; i++) //columns

{

for (int j = 0; j < res.n; j++)

{

double s = 0;

for (int k = 0; k < m1.m; k++)

s += m1.elem[j, k] \* m2.elem[k, i];

res.elem[j, i] = s;

}

}

return res;

}

}

public class SurfaceErmit : Graph3d

{

public Point3d[] points;

Matrix4x4 Mh = new Matrix4x4(4, 4);

Matrix4x4 MhT = new Matrix4x4(4, 4);

bool surface;

public SurfaceErmit(Point3d[] pnts)

{

points = pnts;

surface = true;

double[,] t =

{

{2,-2,1,1 },

{-3,3,-2,-1 },

{0,0,1,0 },

{1,0,0,0}

};

Mh.elem = t;

//transpose

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

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

MhT.elem[i, j] = t[j, i];

}

public void draw(Graphics g, ProjType pt)

{

Matrix4x4 Qx = new Matrix4x4(4, 4);

Matrix4x4 Qy = new Matrix4x4(4, 4);

Matrix4x4 Qz = new Matrix4x4(4, 4);

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

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

{

Qx.elem[i, j] = points[i \* 4 + j].x;

Qy.elem[i, j] = points[i \* 4 + j].y;

Qz.elem[i, j] = points[i \* 4 + j].z;

}

Matrix4x4 T = new Matrix4x4(4, 1);

Matrix4x4 S = new Matrix4x4(1, 4);

double dt = 0.1, ds = 0.1;

List<PointF> pointsf = new List<PointF>();

//draw points

for (double t = 0; t <= 1; t += dt)

{

T.elem[0, 0] = t \* t \* t;

T.elem[1, 0] = t \* t;

T.elem[2, 0] = t;

T.elem[3, 0] = 1;

for (double s = 0; s <= 1; s += ds)

{

S.elem[0, 0] = s \* s \* s;

S.elem[0, 1] = s \* s;

S.elem[0, 2] = s;

S.elem[0, 3] = 1;

Point3d p = new Point3d();

p.x = (S \* Mh \* Qx \* MhT \* T).elem[0, 0];

p.y = (S \* Mh \* Qy \* MhT \* T).elem[0, 0];

p.z = (S \* Mh \* Qz \* MhT \* T).elem[0, 0];

PointF pp = p.project(pt, surface);

g.FillEllipse(Brushes.Black, pp.X, pp.Y, 3, 3);

pointsf.Add(pp);

}

}

for (int t = 0; t <= 10; t++)

{

for (int s = 0; s <= 10; s++)

{

PointF pp = pointsf[t \* 11 + s];

if (t > 0)

{

PointF pb = pointsf[(t - 1) \* 11 + s];

g.DrawLine(Pens.Black, pb, pp);

}

if (s > 0)

{

PointF pl = pointsf[t \* 11 + s - 1];

g.DrawLine(Pens.Black, pl, pp);

}

}

}

}

public void move(int dx, int dy, int dz)

{

var zer = new Point3d(0, 0, 0);

for (int i = 0; i < 2; i++)

{

for (int j = 0; j < 2; j++)

{

int k = i \* 4 + j;

points[k].x += dx;

points[k].y += dy;

points[k].z += dz;

}

}

}

public void rotate(double ax, double ay, double az)

{

Point3d cnt;

var cnt1 = new Point3d(

(points[5].x - points[0].x) / 2 + points[0].x,

(points[5].y - points[0].y) / 2 + points[0].y,

(points[5].z - points[0].z) / 2 + points[0].z

);

var zer = new Point3d(0, 0, 0);

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

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

{

int k = i \* 4 + j;

if (i < 2 && j < 2)

cnt = cnt1;

else

cnt = zer;

if (ax != 0)

{

Point3d p = new Point3d(points[k].x - cnt.x, points[k].y - cnt.y, points[k].z - cnt.z);

points[k] = new Point3d(

cnt.x + p.x,

cnt.y + p.y \* Math.Cos(ax \* Math.PI / 180) - p.z \* Math.Sin(ax \* Math.PI / 180),

cnt.z + p.y \* Math.Sin(ax \* Math.PI / 180) + p.z \* Math.Cos(ax \* Math.PI / 180)

);

}

if (ay != 0)

{

Point3d p = new Point3d(points[k].x - cnt.x, points[k].y - cnt.y, points[k].z - cnt.z);

points[k] = new Point3d(

cnt.x + p.x \* Math.Cos(ay \* Math.PI / 180) - p.z \* Math.Sin(ay \* Math.PI / 180),

cnt.y + p.y,

cnt.z + p.x \* Math.Sin(ay \* Math.PI / 180) + p.z \* Math.Cos(ay \* Math.PI / 180)

);

}

if (az != 0)

{

Point3d p = new Point3d(points[k].x - cnt.x, points[k].y - cnt.y, points[k].z - cnt.z);

points[k] = new Point3d(

cnt.x + p.x \* Math.Cos(az \* Math.PI / 180) - p.y \* Math.Sin(az \* Math.PI / 180),

cnt.y + p.x \* Math.Sin(az \* Math.PI / 180) + p.y \* Math.Cos(az \* Math.PI / 180),

cnt.z + p.z

);

}

}

}

public void scale(double kx, double ky, double kz)

{

Point3d cnt;

Point3d cnt1 = new Point3d((points[5].x - points[0].x) / 2 + points[0].x,

(points[5].y - points[0].y) / 2 + points[0].y,

(points[5].z - points[0].z) / 2 + points[0].z);

var zer = new Point3d(0, 0, 0);

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

{

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

{

int i = 4 \* k + j;

if (k < 2 && j < 2)

cnt = cnt1;

else

cnt = zer;

Point3d p = new Point3d(points[i].x - cnt.x, points[i].y - cnt.y, points[i].z - cnt.z);

points[i] = new Point3d(

cnt.x + p.x \* kx,

cnt.y + p.y \* ky,

cnt.z + p.z \* kz

);

}

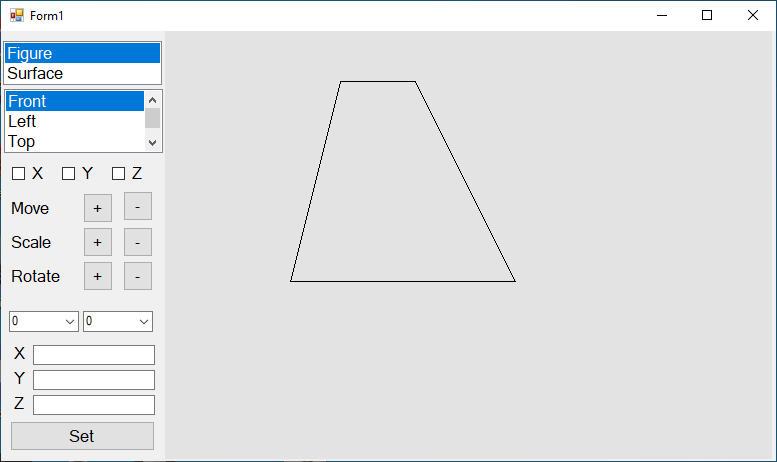
}

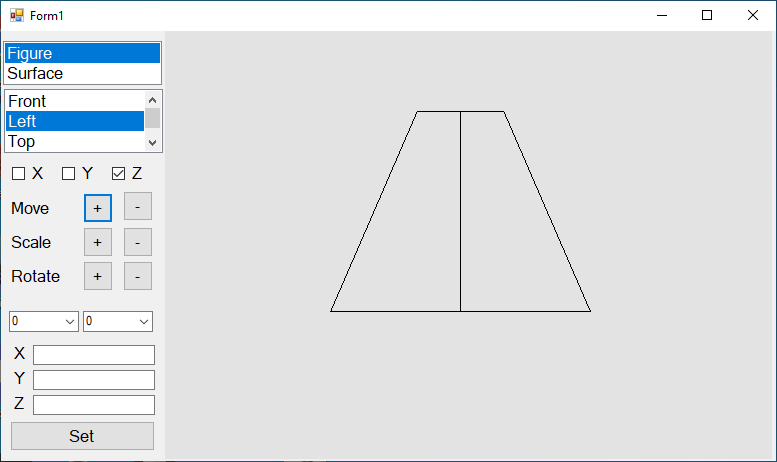
}

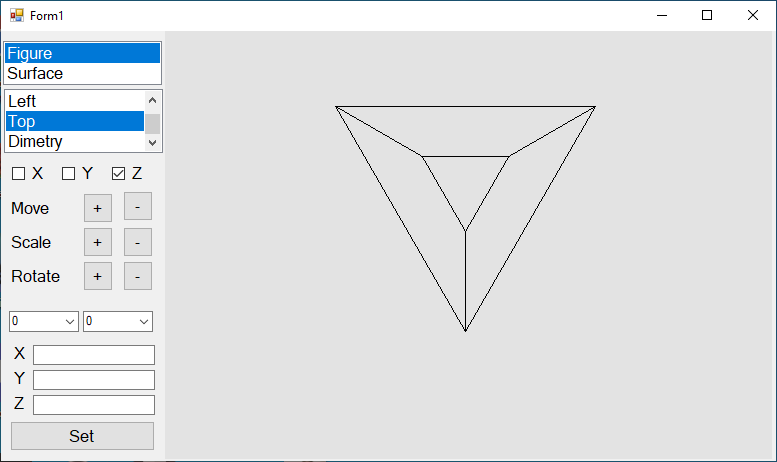
}

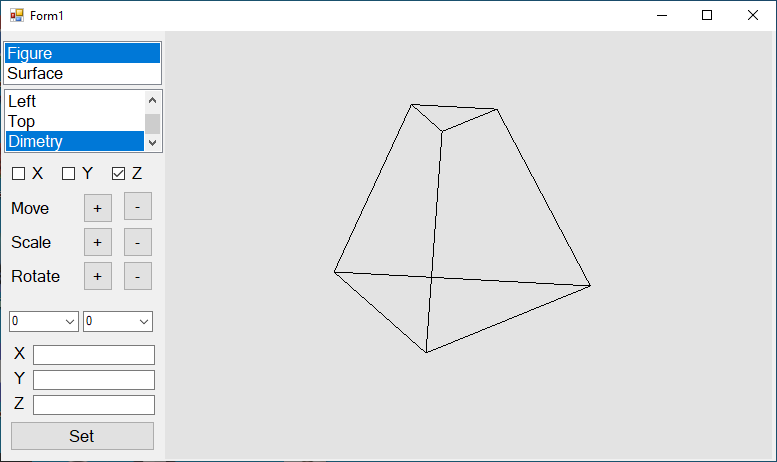
}

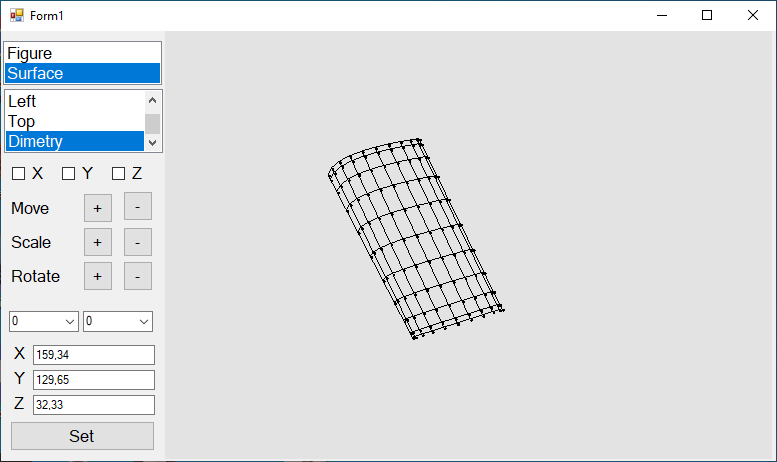
**Результат роботи програми**

****

****

****

****

****

**Висновок**

Під час виконання лабораторної роботи працював з тривимірною графікою, були реалізовані афінні перетворення для 3D об’єктів. Також познайомився з різними видами проекцій такі як диметрія та ізометрія.