Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра автоматизации обработки информации (АОИ)

«Проективные преобразования 3D изображений»

Отчет о выполнении лабораторной работы по дисциплине "Компьютерная графика"

Вариант № 3

Выполнил:

Студент гр. 428-1

Ван А.Г.

Проверил:

Доцент каф. АОИ ТУСУР,

канд. тех. наук

Т.О. Перемитина

**Содержание**

1. [Введение…………………………………………………………………....3](#_Введение)
2. [Основная часть……………………………………………………………..4](#_Основная_часть)
3. [Заключение…………………………………………………………………6](#_Заключение )
4. [Приложение………………………………………………………………...7](#_Приложение )

# Введение

**Цель работы:** Применение проективных преобразовани.

**Задание:** Построить 3D модель фигуры (согласно варианту задания) и выполнить проективное преобразование заданной фигуры.

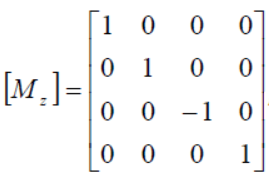
Вариант № 3:

Реализуйте программу, позволяющую выполнять композицию преобразований над четырехгранной пирамидой: отражение относительно различных плоскостей и аксонометрическое триметрическое проецирование.

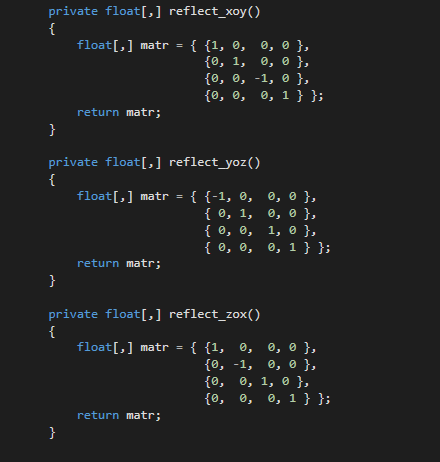
# Основная часть

В качестве среды для разработки выбрана интегрированная среда программирования Microsoft Visual Studio.

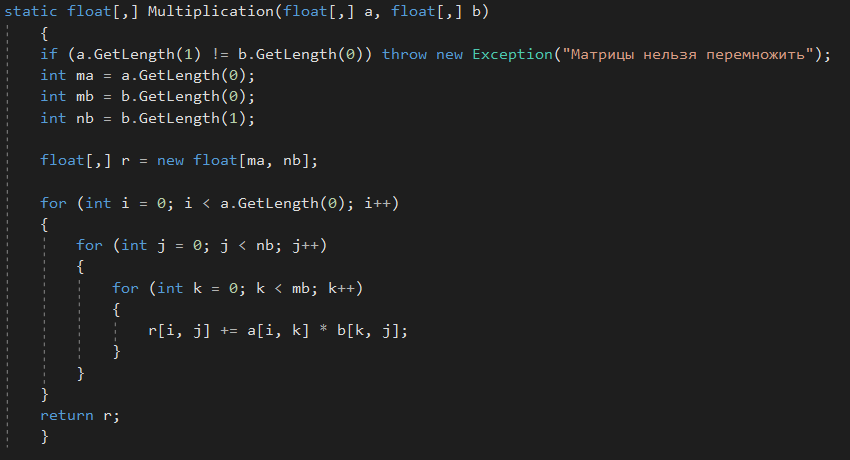
Для реализации отражения фигуры относительно плоскости XOY используем матрицу преобразования:



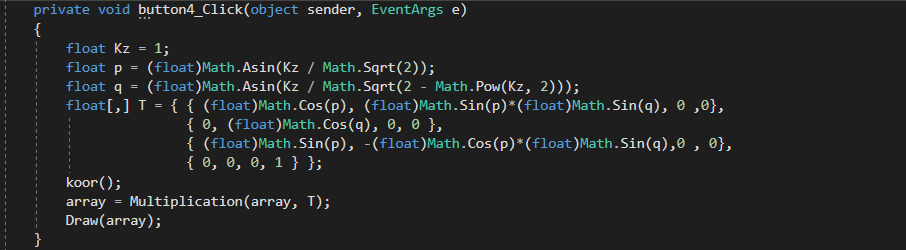
Для отражений относительно других плоскостей, соответствующие матрицы:



Для умножения матриц мы используем функцию Multiplication(float[,] a, float[,] b)



Построение аксонометрического триметрического проецирования:



# Заключение

Реализована программа с функциональными возможностями:

* построение осей координат *XYZ* и их подписи;
* построение фигуры, согласно варианту;
* отражение фигуры, относительно плоскостей.
* построение аксонометрического триметрического проецирования фигуры.

# Приложение

Результаты работы:

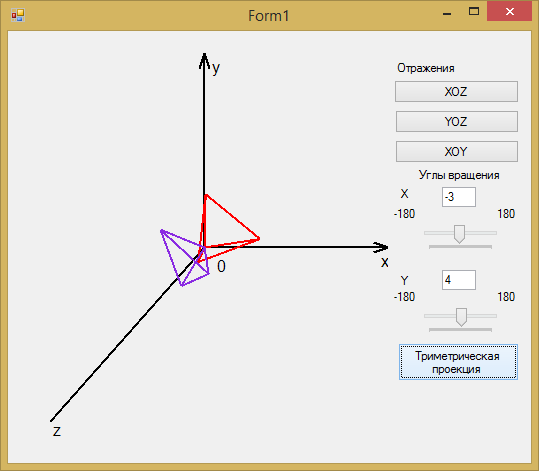


Рисунок . Триметрическая проекция

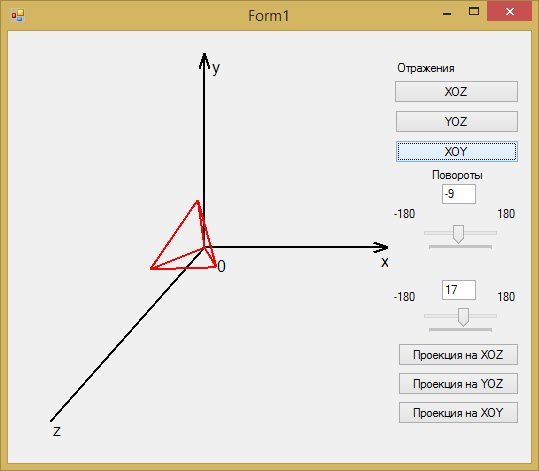


Рисунок . Отражение фигуры относительно плоскости XOY

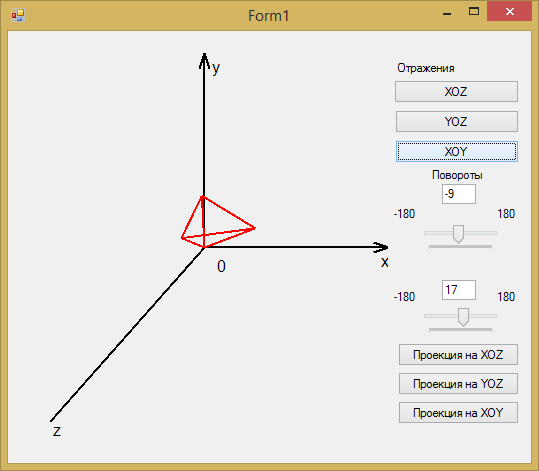


Рисунок 3. Отражение фигуры относительно плоскости XOY

Листинг программы

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 WindowsFormsApplication1

{

struct PointF3d

{

public PointF3d(float x, float y, float z)

{

this.x = x;

this.y = y;

this.z = z;

}

public float x;

public float y;

public float z;

}

public partial class Form1 : Form

{

Graphics g;

PointF center;

float Q1 ;

float Q2;

float[,] figure = { { 50, 0, 0, 1 },

{ 0, 50, 0, 1 },

{ 0, 0, 50, 1 },

{ 0, 0, 0, 1 } };

public Form1()

{

InitializeComponent();

g = display.CreateGraphics();

center = new PointF(display.Width / 2, display.Height / 2);

}

private void draw\_axises()

{

float width = display.Width;

float height = display.Height;

Pen pen = new Pen(Color.Black, 2);

g.DrawLine(pen, center.X, center.Y, width, center.Y); // x axis

g.DrawLine(pen, width, center.Y, width - 15, center.Y - 5);

g.DrawLine(pen, width, center.Y, width - 15, center.Y + 5);

g.DrawLine(pen, center.X, 10, center.X, center.Y); // y axis

g.DrawLine(pen, center.X, 10, center.X - 5, 25);

g.DrawLine(pen, center.X, 10, center.X + 5, 25);

g.DrawLine(pen, center.X, center.Y, 30, height - 30);

Font mf = new Font("Arial", 12);

g.DrawString("x", mf, Brushes.Black, new PointF( width - 10, center.Y + 5 ));

g.DrawString("y", mf, Brushes.Black, new PointF( center.X + 5, 15));

g.DrawString("z", mf, Brushes.Black, new PointF(30, height - 30));

g.DrawString("0", mf, Brushes.Black, new PointF(center.X + 10, center.Y + 10) );

}

private float[,] reflect\_xoy()

{

float[,] matr = { {1, 0, 0, 0 },

{0, 1, 0, 0 },

{0, 0, -1, 0 },

{0, 0, 0, 1 } };

return matr;

}

private float[,] reflect\_yoz()

{

float[,] matr = { {-1, 0, 0, 0 },

{ 0, 1, 0, 0 },

{ 0, 0, 1, 0 },

{ 0, 0, 0, 1 } };

return matr;

}

private float[,] reflect\_zox()

{

float[,] matr = { {1, 0, 0, 0 },

{0, -1, 0, 0 },

{0, 0, 1, 0 },

{0, 0, 0, 1 } };

return matr;

}

private float[,] multiply(float [,] m1, float[,] m2)

{

int columns = m1.GetLength(1);

int rows = m1.GetLength(0);

float[,] res = new float[rows, columns];

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

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

{

for (int loop = 0; loop < columns; loop++)

res[i, j] += m1[i, loop] \* m2[loop, j];

}

return res;

}

private void draw\_figure()

{

Pen pen = new Pen(Color.Red, 2);

// [ X ], [ Y ] [ X ] [ Y ]

g.DrawLine(pen, center.X + figure[0, 0] - figure[0, 2] / 2, center.Y - figure[0, 1] + figure[0, 2] / 2, center.X + figure[1, 0] - figure[1, 2] / 2, center.Y - figure[1, 1] + figure[1, 2] / 2);

g.DrawLine(pen, center.X + figure[1, 0] - figure[1, 2] / 2, center.Y - figure[1, 1] + figure[1, 2] / 2, center.X + figure[2, 0] - figure[2, 2] / 2, center.Y - figure[2, 1] + figure[2, 2] / 2);

g.DrawLine(pen, center.X + figure[2, 0] - figure[2, 2] / 2, center.Y - figure[2, 1] + figure[2, 2] / 2, center.X + figure[0, 0] - figure[0, 2] / 2, center.Y - figure[0, 1] + figure[0, 2] / 2);

g.DrawLine(pen, center.X + figure[0, 0] - figure[0, 2] / 2, center.Y - figure[0, 1] + figure[0, 2] / 2, center.X + figure[3, 0] - figure[3, 2] / 2, center.Y - figure[3, 1] + figure[3, 2] / 2);

g.DrawLine(pen, center.X + figure[1, 0] - figure[1, 2] / 2, center.Y - figure[1, 1] + figure[1, 2] / 2, center.X + figure[3, 0] - figure[3, 2] / 2, center.Y - figure[3, 1] + figure[3, 2] / 2);

g.DrawLine(pen, center.X + figure[2, 0] - figure[2, 2] / 2, center.Y - figure[2, 1] + figure[2, 2] / 2, center.X + figure[3, 0] - figure[3, 2] / 2, center.Y - figure[3, 1] + figure[3, 2] / 2);

}

private void btn\_reflect\_zox\_Click(object sender, EventArgs e)

{

Refresh();

draw\_axises();

var ref\_matr = reflect\_zox();

this.figure = this.multiply(figure, ref\_matr);

draw\_figure();

}

private void btn\_reflect\_xoy\_Click(object sender, EventArgs e)

{

Refresh();

draw\_axises();

var ref\_matr = reflect\_xoy();

this.figure = this.multiply(figure, ref\_matr);

draw\_figure();

}

private void btn\_reflect\_yoz\_Click(object sender, EventArgs e)

{

Refresh();

draw\_axises();

var ref\_matr = reflect\_yoz();

this.figure = this.multiply(figure, ref\_matr);

draw\_figure();

}

private float[,] reflect\_vrash()

{

float[,] matr = { { 1, 0, 0, 0 },

{0, (float)Math.Cos( Q2\* Math.PI / 180), (float)Math.Sin( Q2\*Math.PI / 180), 0 },

{0, (float)-Math.Sin( Q2\*Math.PI / 180), (float)Math.Cos( Q2\* Math.PI / 180), 0 },

{0, 0, 0, 1 } };

return matr;

}

private void trackBar1\_Scroll(object sender, EventArgs e) //по иксу

{

Refresh();

draw\_axises();

Q2 = 0;

Q2 = trackBar1.Value;

Edit2.Text = Q2.ToString();

var ref\_matr = reflect\_vrash();

this.figure = this.multiply(figure, ref\_matr);

draw\_figure();

}

private float[,] reflect\_vresh()

{

float[,] matr = { { (float)Math.Cos(Q1\* Math.PI / 180), 0, (float)-Math.Sin( Q1 \* Math.PI / 180), 0 },

{0, 1, 0, 0 },

{(float)Math.Sin( Q1\*Math.PI / 180), 0, (float)Math.Cos( Q1\* Math.PI / 180), 0 },

{0, 0, 0, 1 } };

return matr;

}

private void redraw(float[,] matr)

{

Pen pen = new Pen(Color.BlueViolet, 2);

// [ X ], [ Y ] [ X ] [ Y ]

g.DrawLine(pen, center.X + matr[0, 0] - matr[0, 2] / 2, center.Y - matr[0, 1] + matr[0, 2] / 2, center.X + matr[1, 0] - matr[1, 2] / 2, center.Y - matr[1, 1] + matr[1, 2] / 2);

g.DrawLine(pen, center.X + matr[1, 0] - matr[1, 2] / 2, center.Y - matr[1, 1] + matr[1, 2] / 2, center.X + matr[2, 0] - matr[2, 2] / 2, center.Y - matr[2, 1] + matr[2, 2] / 2);

g.DrawLine(pen, center.X + matr[2, 0] - matr[2, 2] / 2, center.Y - matr[2, 1] + matr[2, 2] / 2, center.X + matr[0, 0] - matr[0, 2] / 2, center.Y - matr[0, 1] + matr[0, 2] / 2);

g.DrawLine(pen, center.X + matr[0, 0] - matr[0, 2] / 2, center.Y - matr[0, 1] + matr[0, 2] / 2, center.X + matr[3, 0] - matr[3, 2] / 2, center.Y - matr[3, 1] + matr[3, 2] / 2);

g.DrawLine(pen, center.X + matr[1, 0] - matr[1, 2] / 2, center.Y - matr[1, 1] + matr[1, 2] / 2, center.X + matr[3, 0] - matr[3, 2] / 2, center.Y - matr[3, 1] + matr[3, 2] / 2);

g.DrawLine(pen, center.X + matr[2, 0] - matr[2, 2] / 2, center.Y - matr[2, 1] + matr[2, 2] / 2, center.X + matr[3, 0] - matr[3, 2] / 2, center.Y - matr[3, 1] + matr[3, 2] / 2);

}

private void trackBar2\_Scroll(object sender, EventArgs e) //по игрику

{

Refresh();

draw\_axises();

Q1 = 0;

Q1 = trackBar2.Value;

Edit1.Text = Q1.ToString();

var ref\_matr = reflect\_vresh();

this.figure = this.multiply(figure, ref\_matr);

draw\_figure();

}

private float[,] projection\_trimetry(float alpha, float beta)

{

float cos\_beta = (float)Math.Cos(beta);

float sin\_beta = (float)Math.Sin(beta);

float cos\_alpha = (float)Math.Cos(alpha);

float sin\_alpha = (float)Math.Sin(alpha);

float[,] matr = { { cos\_beta, sin\_alpha \* sin\_beta, 0, 0},

{ 0, cos\_alpha, 0, 0},

{ sin\_beta, -sin\_alpha \* cos\_beta, 0, 0},

{ 0, 0, 0, 1 } };

return matr;

}

private void button2\_Click(object sender, EventArgs e)

{

float angle\_x = float.Parse(Edit1.Text);

float angle\_y = float.Parse(Edit2.Text);

var ref\_matr = projection\_trimetry(angle\_x, angle\_y);

ref\_matr = this.multiply(figure, ref\_matr);

redraw(ref\_matr);

}

}

}