Учреждение образования

«Гомельский Государственный технический университет имени П.О.Сухого»

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

Отчет по лабораторной работе №2

«Реализация программ построения и преобразования трехмерных объектов»

По дисциплине «Алгоритмические основы современной компьютерной графики»

Выполнил студент гр.ИТЗ-21

Касаткин Д.О.

Проверил преподаватель

Комракова М.В.

Гомель 2019

**Цель работы:** изучить геометрические преобразования трехмерных объектов.

**Задание.** Для выбранного варианта необходимо: 1. Создать матрицу ключевых точек для каждой фигуры, образующей замкнутый контур. По координатам ключевых точек построить изображение заданной фигуры, используя 3D график. 2. Провести перенос фигуры на расстояние вдоль горизонтали и вертикали. Построить изображение преобразованной фигуры. 3. Провести неоднородное и однородное масштабирование заданной фигуры. Построить изображение преобразованной фигуры. 4. Провести поворот заданной фигуры. Построить изображение преобразованной фигуры.

Листинг программы:

using SharpGL;

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 Lab\_2

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

comboBox1.Items.AddRange(new string[] { "X", "Y", "Z" });

comboBox1.DropDownStyle = System.Windows.Forms.ComboBoxStyle.DropDownList;

comboBox1.SelectedIndex = 0;

create\_osxy();

drow\_figures();

}

List<Osxy> osxy\_rhombus = new List<Osxy>();

List<Osxy> kub = new List<Osxy>();

private void create\_osxy()

{

osxy\_rhombus.Add(new Osxy(0f, 6f,0f));

osxy\_rhombus.Add(new Osxy(3f, 0f,0f));

osxy\_rhombus.Add(new Osxy(0f, -6f,0f));

osxy\_rhombus.Add(new Osxy(-3f, 0f,0f));

kub.Add(new Osxy(2f,2f,0f));

kub.Add(new Osxy(2f, -2f, 0f));

kub.Add(new Osxy(-2f,-2f, 0f));

kub.Add(new Osxy(-2f, 2f, 0f));

}

Single angle\_x = 0, angle\_y = 0, angle\_z = 0;

Single matrix\_x = 0,matrix\_y = 0,matrix\_z = 0;

private void drow\_figures()

{

OpenGL gl = this.openGLControl1.OpenGL;

// Очистка экрана и буфера глубин

gl.Clear(OpenGL.GL\_COLOR\_BUFFER\_BIT | OpenGL.GL\_DEPTH\_BUFFER\_BIT);

gl.Enable(OpenGL.GL\_DEPTH\_TEST);

// Сбрасываем модельно-видовую матрицу

gl.PushMatrix();

gl.Translate(matrix\_x, matrix\_y, matrix\_z - 20.0f);

gl.Rotate(angle\_x, 1, 0, 0);

gl.Rotate(angle\_y, 0, 1, 0);

gl.Rotate(angle\_z, 0, 0, 1);

gl.Translate(0.0f, 0.0f, 2.5f);

gl.PolygonMode(OpenGL.GL\_FRONT\_AND\_BACK, OpenGL.GL\_LINE);

gl.Begin(OpenGL.GL\_POLYGON);

gl.Color(1f, 1f, 0f);

//рисуем левый прямоугольник

foreach (Osxy figures in osxy\_rhombus)

{

gl.Vertex(figures.x, figures.y, figures.z);

}

gl.End();

gl.Color(1f, 0f, 0f);

foreach (Osxy figures in osxy\_rhombus)

{

gl.Vertex(figures.x, figures.y, figures.z-5);

}

gl.End();

for (int i1 = 0; i1 < osxy\_rhombus.Count - 1; i1++)

{

gl.Begin(OpenGL.GL\_POLYGON);

gl.Color(1f, 0f, 0f); // красный цвет

gl.Vertex(osxy\_rhombus[i1].x, osxy\_rhombus[i1].y, osxy\_rhombus[i1].z);

gl.Vertex(osxy\_rhombus[i1 + 1].x, osxy\_rhombus[i1 + 1].y, osxy\_rhombus[i1 + 1].z);

gl.Vertex(osxy\_rhombus[i1 + 1].x, osxy\_rhombus[i1 + 1].y, osxy\_rhombus[i1 + 1].z-5);

gl.Vertex(osxy\_rhombus[i1].x, osxy\_rhombus[i1].y, osxy\_rhombus[i1].z-5);

gl.End();

}

gl.Begin(OpenGL.GL\_POLYGON);

gl.Color(1f, 0f, 0f); // красный цвет

gl.Vertex(osxy\_rhombus[0].x, osxy\_rhombus[0].y, osxy\_rhombus[0].z);

gl.Vertex(osxy\_rhombus[3].x, osxy\_rhombus[3].y, osxy\_rhombus[3].z);

gl.Vertex(osxy\_rhombus[3].x, osxy\_rhombus[3].y, osxy\_rhombus[3].z-5);

gl.Vertex(osxy\_rhombus[0].x, osxy\_rhombus[0].y, osxy\_rhombus[0].z-5);

gl.End();

gl.PolygonMode(OpenGL.GL\_FRONT\_AND\_BACK, OpenGL.GL\_LINE);

gl.Begin(OpenGL.GL\_POLYGON);

gl.Color(1f, 0f, 0f);

//рисуем левый прямоугольник

foreach (Osxy figures in kub)

{

gl.Vertex(figures.x, figures.y, figures.z);

}

gl.End();

gl.Begin(OpenGL.GL\_POLYGON);

// Указываем цвет вершин

gl.Color(1f, 0f, 0f);

foreach (Osxy figures in kub)

{

gl.Vertex(figures.x, figures.y, figures.z-5);

}

gl.End();

for (int i1 = 0; i1 < kub.Count - 1; i1++)

{

gl.Begin(OpenGL.GL\_POLYGON);

gl.Color(1f, 0f, 0f); // красный цвет

gl.Vertex(kub[i1].x, kub[i1].y, kub[i1].z);

gl.Vertex(kub[i1 + 1].x, kub[i1 + 1].y, kub[i1 + 1].z);

gl.Vertex(kub[i1 + 1].x, kub[i1 + 1].y, kub[i1 + 1].z-5);

gl.Vertex(kub[i1].x, kub[i1].y, kub[i1].z-5);

gl.End();

}

gl.Begin(OpenGL.GL\_POLYGON);

gl.Color(1f, 0f, 0f); // красный цвет

gl.Vertex(kub[0].x, kub[0].y, kub[0].z-5);

gl.Vertex(kub[3].x, kub[3].y, kub[3].z-5);

gl.Vertex(kub[3].x, kub[3].y, kub[3].z-5);

gl.Vertex(kub[0].x, kub[0].y, kub[0].z-5);

gl.End();

gl.PopMatrix();

}

private void Button1\_Click(object sender, EventArgs e)

{

drow\_figures();

}

// построение матрицы вращения

private void Button2\_Click(object sender, EventArgs e)

{

drow\_figures();

}

private void button3\_Click(object sender, EventArgs e)

{

if (comboBox1.SelectedIndex == 0)

matrix\_x++;

if (comboBox1.SelectedIndex == 1)

matrix\_y++;

if (comboBox1.SelectedIndex == 2)

matrix\_z++;

drow\_figures();

}

private void button4\_Click(object sender, EventArgs e)

{

if (comboBox1.SelectedIndex == 0)

angle\_x+=10;

if (comboBox1.SelectedIndex == 1)

angle\_y += 10;

if (comboBox1.SelectedIndex == 2)

angle\_z += 10;

drow\_figures();

}

private void button6\_Click(object sender, EventArgs e)

{

if (comboBox1.SelectedIndex == 0)

matrix\_x--;

if (comboBox1.SelectedIndex == 1)

matrix\_y--;

if (comboBox1.SelectedIndex == 2)

matrix\_z--;

drow\_figures();

}

private void radioButton2\_CheckedChanged(object sender, EventArgs e)

{

drow\_figures();

}

private double degrees(int degrees)

{

return Math.PI \* degrees / 180.0;

}

private void button5\_Click(object sender, EventArgs e)

{

if (comboBox1.SelectedIndex == 0)

angle\_x -= 10;

if (comboBox1.SelectedIndex == 1)

angle\_y -= 10;

if (comboBox1.SelectedIndex == 2)

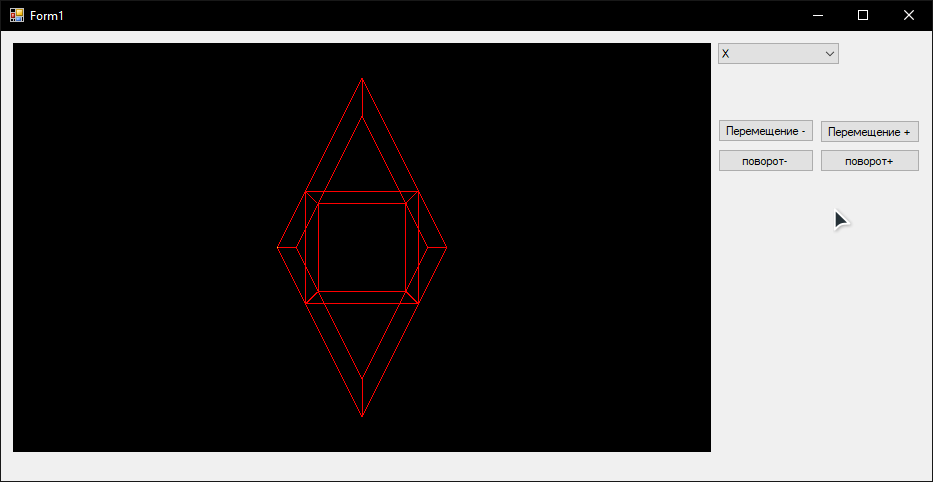
angle\_z -= 10;

drow\_figures();

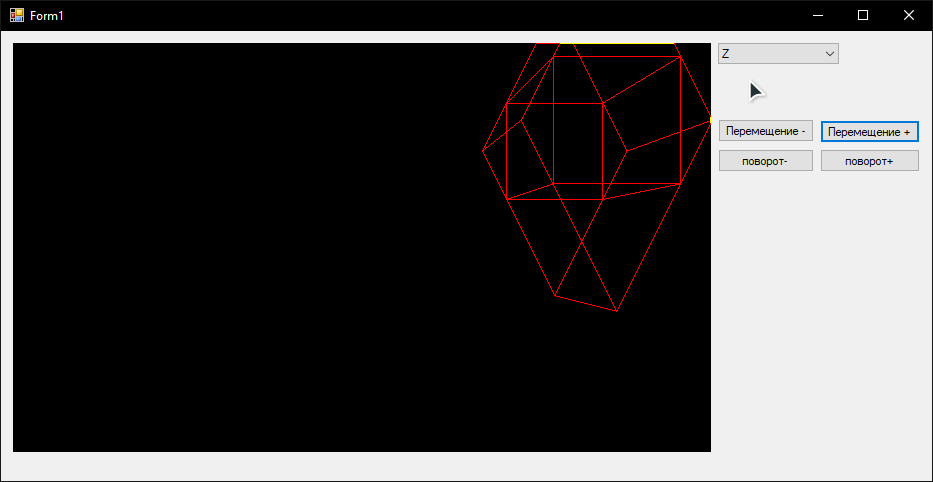
}

}

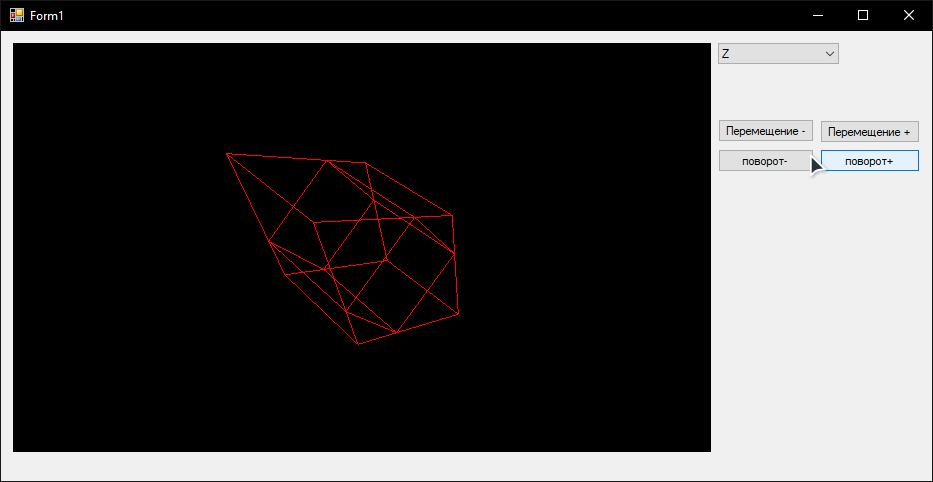
}



Скриншот 1 : запуск программы.



Скриншот 2 : перемещение по осям.



Скриншот 3: поворот фигуры.

**Вывод:** в ходе выполнения работы провели геометрические преобразования трёхмерныхобъектов: масштабирование, перемещение, поворот.