МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
КАФЕДРА САПР

ОТЧЕТ

по лабораторной работе №5

по дисциплине «Компьютерная графика»

Тема: «Исследование алгоритмов видимости сложных сцен»

Вариант 1

Студенты гр. 9309 _____ Аль Сайед А.З.

_____ Серов А.В.

_____ Юшин Е.В.

Преподаватель _____ Матвеева И.В.

Санкт-Петербург
2022

Цель работы

Реализовать алгоритм выявления видимости сложных сцен.

Задание

Обеспечить реализацию алгоритма выявления видимых граней и ребер для одиночного выпуклого объемного тела.

Математическая модель

Видимость грани определяется пересечением ее плоскости отрезка, концами которого являются точка наблюдения и одна внутренняя точка фигуры.

Внутренняя точка берется как среднее арифметическое всех вершин по каждой координате.

Точка наблюдения первоначально задается в координате (6, 6, 6), далее изменяется с помощью клавиш q, w, e, a, s, d.

Для определения факта пересечения используется каноническая форма плоскости и параметрическая запись прямой.

Из уравнения прямой

$$f(x,y,z) = \begin{vmatrix} x - x_1 & y - y_1 & z - z_1 \\ x_2 - x_1 & y_2 - y_1 & z_2 - z_1 \\ x_3 - x_1 & y_3 - y_1 & z_3 - z_1 \end{vmatrix}$$

выводится уравнение вида

$$ax + by + cz + d = 0, \text{где}$$

коэффициенты

$$a = (y_2 - y_1) * (z_3 - z_1) - (z_2 - z_1) * (y_3 - y_1),$$
$$b = -(x_2 - x_1) * (z_3 - z_1) - (z_2 - z_1) * (x_3 - x_1),$$
$$c = (x_2 - x_1) * (y_3 - y_1) - (y_2 - y_1) * (x_3 - x_1),$$
$$d = -x_1 * a - y_1 * b - z_1 * c.$$

Далее по известным нам координатам точки наблюдения и внутренней точки, а также коэффициентам a, b, c и d вычисляется значение t.

$$\begin{matrix} ax + by + cz + d = 0 \\ \overline{P(t)} = \overline{P_1} + (\overline{P_2} - \overline{P_1}) * t \end{matrix} => t = \frac{-d * a * x_1 - b * y_1 - c * z_1}{a * (x_2 - x_1) + b * (y_2 - y_1) + c * (z_2 - z_1)}$$

Контрольный пример

Контрольные примеры представлены на рис. 1, 2 и 3.

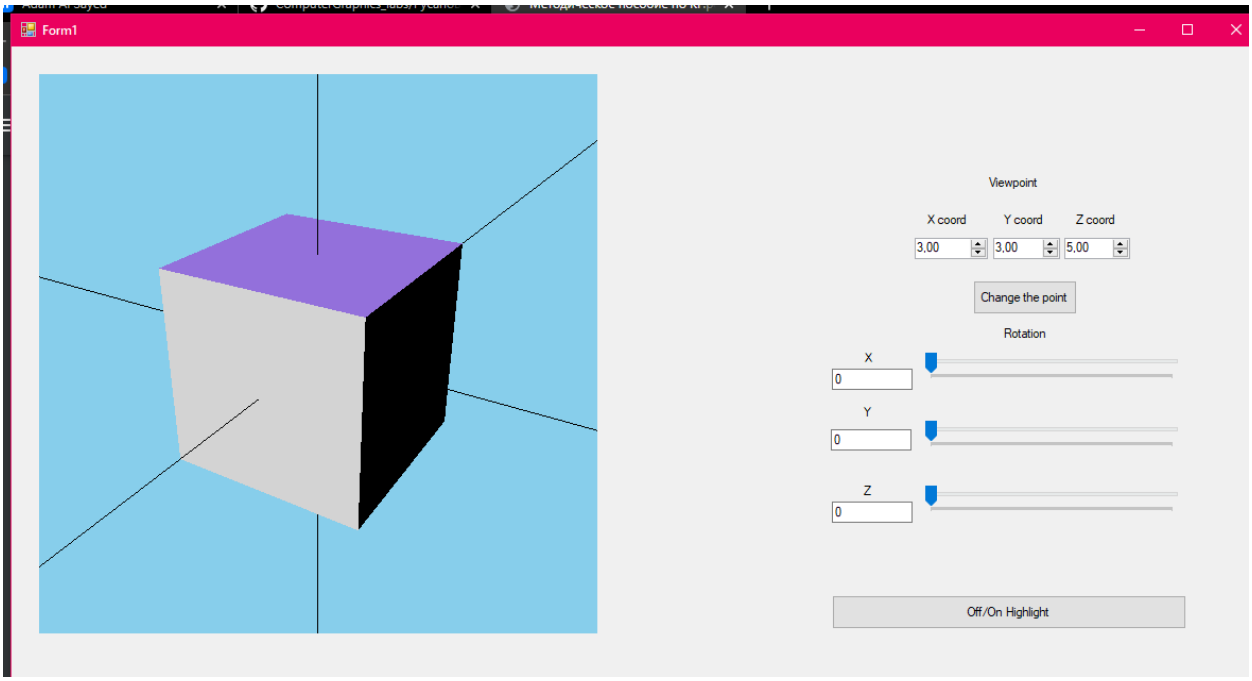

Рис.1. Передние грани куба

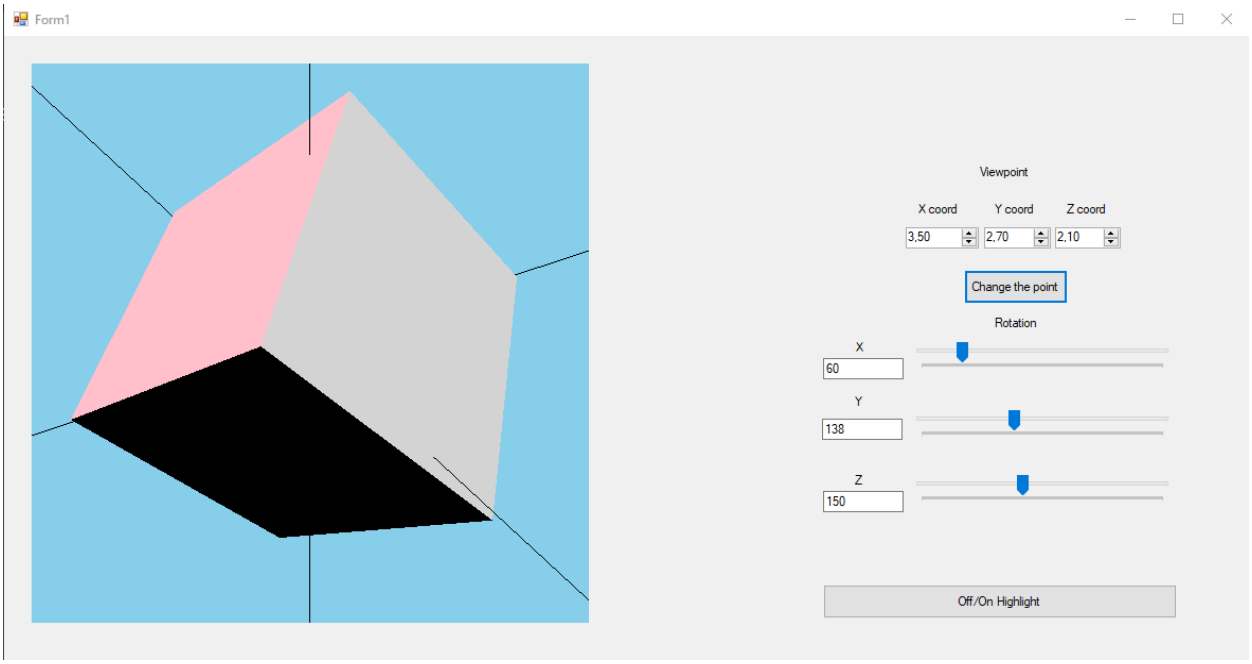

Рис. 2. Обеспечение поворота куба и изменение точки обзора

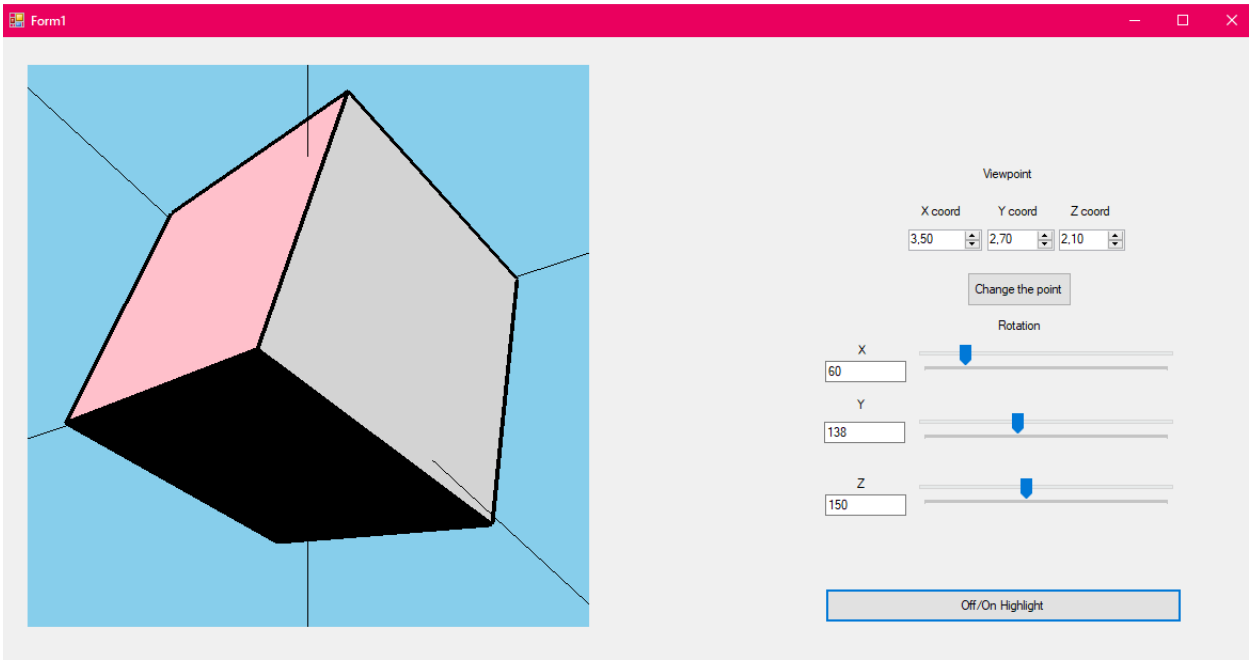

Рис. 3. Выделение видимых граней куба

Код программы

```
using OpenTK.Graphics;
using OpenTK.Graphics.OpenGL;
using System;
using System.Windows.Media.Media3D;

namespace CG_lab5
{
    public class Cube
    {
            public Point3D[] cubeVertex;
            public Color4[] color;
            public bool[] vertexVisibility = new bool[24];
            public bool[] ribsVisibility = new bool[24];


            public Cube()
        {
                cubeVertex = new Point3D[24];
                color = new Color4[8];

                cubeVertex[0] = new Point3D(1.0f, -1.0f, -1.0f);
                cubeVertex[1] = new Point3D(1.0f, -1.0f, 1.0f);
                cubeVertex[2] = new Point3D(1.0f, 1.0f, 1.0f);
                cubeVertex[3] = new Point3D(1.0f, 1.0f, -1.0f);

                cubeVertex[4] = new Point3D(-1.0f, -1.0f, -1.0f);
                cubeVertex[5] = new Point3D(-1.0f, -1.0f, 1.0f);
                cubeVertex[6] = new Point3D(-1.0f, 1.0f, 1.0f);
                cubeVertex[7] = new Point3D(-1.0f, 1.0f, -1.0f);

                cubeVertex[8] = new Point3D(-1.0f, -1.0f, -1.0f);
                cubeVertex[9] = new Point3D(1.0f, -1.0f, -1.0f);
                cubeVertex[10] = new Point3D(1.0f, -1.0f, 1.0f);
                cubeVertex[11] = new Point3D(-1.0f, -1.0f, 1.0f);

                cubeVertex[12] = new Point3D(1.0f, 1.0f, -1.0f);
                cubeVertex[13] = new Point3D(1.0f, -1.0f, -1.0f);
                cubeVertex[14] = new Point3D(-1.0f, -1.0f, -1.0f);
                cubeVertex[15] = new Point3D(-1.0f, 1.0f, -1.0f);

                cubeVertex[16] = new Point3D(-1.0f, -1.0f, 1.0f);
                cubeVertex[17] = new Point3D(-1.0f, 1.0f, 1.0f);
                cubeVertex[18] = new Point3D(1.0f, 1.0f, 1.0f);
                cubeVertex[19] = new Point3D(1.0f, -1.0f, 1.0f);

                cubeVertex[20] = new Point3D(1.0f, 1.0f, 1.0f);
                cubeVertex[21] = new Point3D(-1.0f, 1.0f, 1.0f);
                cubeVertex[22] = new Point3D(-1.0f, 1.0f, -1.0f);
                cubeVertex[23] = new Point3D(1.0f, 1.0f, -1.0f);

                initColors();
            }

            public Cube(Point3D[] cubeArray)
        {
                cubeVertex = cubeArray;
                initColors();
            }

            public Cube(Cube cube)
        {
                cube.vertexVisibility.CopyTo(vertexVisibility, 0);
                cube.ribsVisibility.CopyTo(ribsVisibility, 0);
```

```
                initColors();
        }

            private void initColors()
            {
                color = new Color4[6];
                color[0] = Color4.Black;
                color[1] = Color4.Blue;
                color[2] = Color4.Pink;
                color[3] = Color4.Red;
                color[4] = Color4.LightGray;
                color[5] = Color4.MediumPurple;
            }

            public void Draw()
        {
                GL.Begin(BeginMode.Quads);

                for (int i = 0; i < 6; i++)
            {
                    GL.Color4(color[i]);
                    for (int j = 0; j < 4; j++)
                        GL.Vertex3(cubeVertex[4*i + j].X, cubeVertex[4 * i +
j].Y, cubeVertex[4 * i + j].Z);
                }
                GL.End();
            }

            public void CountVisible(Point3D pointView)
            {
                double distance = 0;
                for (int i = 0; i < 24; ++i)
                {
                    if (FindDistance(pointView, cubeVertex[i]) > distance)
                    {
                        distance = FindDistance(pointView, cubeVertex[i]);
                    }
                }
                for (int i = 0, vertex = 0; i < 24; i += 4, ++vertex)
                {
                    if (ApproxEqual(FindDistance(pointView, cubeVertex[i]),
distance)
                        || ApproxEqual(FindDistance(pointView, cubeVertex[i
+ 1]), distance)
                        || ApproxEqual(FindDistance(pointView, cubeVertex[i
+ 2]), distance)
                        || ApproxEqual(FindDistance(pointView, cubeVertex[i
+ 3]), distance))
                        vertexVisibility[vertex] = false;
                    else
                        vertexVisibility[vertex] = true;
                }
                for (int i = 0, ribs = 0; i < 24; i += 2, ++ribs)
                {
                    if (ApproxEqual(FindDistance(pointView, cubeVertex[i]),
distance)
                        || ApproxEqual(FindDistance(pointView, cubeVertex[i
+ 1]), distance))
                        ribsVisibility[ribs] = false;
                    else
                        ribsVisibility[ribs] = true;
                }

            }
```

```
            double FindDistance(Point3D from, Point3D to)
        {
                return Math.Sqrt(Math.Pow(from.X - to.X, 2) + Math.Pow(from.Y -
to.Y, 2) + Math.Pow(from.Z - to.Z, 2));
        }

            public void DrawVisible(bool showVisible = true)
        {
                GL.LineWidth(4);
                GL.Begin(BeginMode.Quads);

                for (int i = 0; i < 6; i++)
                {
                      GL.Color4(color[i]);
                      if (vertexVisibility[i] == showVisible)
                            for (int j = 0; j < 4; j++)
                                  GL.Vertex3(cubeVertex[4 * i + j].X,
cubeVertex[4 * i + j].Y, cubeVertex[4 * i + j].Z);
                }
                GL.End();

                GL.Color4(Color4.Black);
                for (int i = 0, ribs = 0; i < 24; i += 2, ribs++)
                {
                      if (ribsVisibility[ribs] == showVisible)
                      {
                            GL.Begin(BeginMode.Lines);
                            GL.Vertex3(cubeVertex[i].X - 0.01, cubeVertex[i].Y -
0.01, cubeVertex[i].Z - 0.01);
                            GL.Vertex3(cubeVertex[i+1].X - 0.01,
cubeVertex[i+1].Y - 0.01, cubeVertex[i+1].Z - 0.01);
                            GL.Vertex3(cubeVertex[i].X + 0.01, cubeVertex[i].Y +
0.01, cubeVertex[i].Z + 0.01);
                            GL.Vertex3(cubeVertex[i+1].X + 0.01,
cubeVertex[i+1].Y + 0.01, cubeVertex[i+1].Z + 0.01);
                            GL.End();
                      }
                }
                GL.LineWidth(1);
            }
            bool ApproxEqual(double double1, double double2)
            {
                double difference = Math.Abs(double1 * .00001);
                if (Math.Abs(double1 - double2) <= difference)
                      return true;
                else return false;
            }
      }
}
```

Вывод

В ходе выполнения лабораторной работы были получены навыки отрисовки объемных объектов с использованием алгоритма выявления видимости граней и ребер одиночного выпуклого тела.