**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. 9301 |  | Примакова Е.Е. |
|  |  | Русанова К.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

выводится уравнение вида

коэффициенты

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

**Контрольный пример**

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

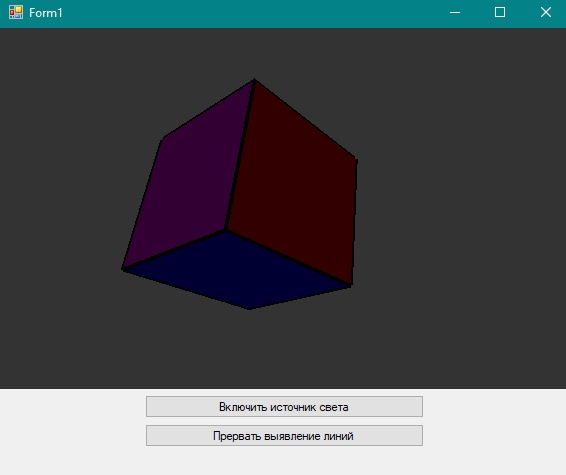


Рис.1. Передние грани куба

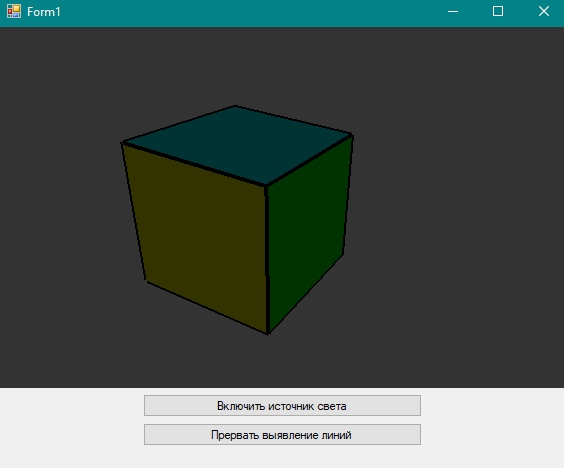


Рис. 2. передние грани куба

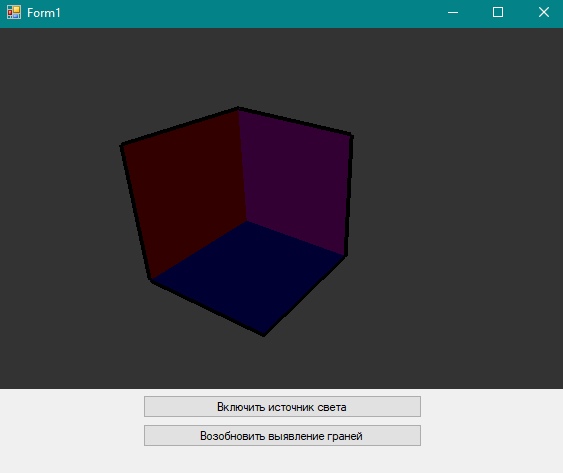


Рис. 3. Куб с убранными гранями

На 1 рисунке показан куб с передней стороны, на 2 рисунке показан этот же куб с задней стороны, на 3 рисунке убраны грани, показанные на 2 рисунке.

**Код 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;

using SharpGL;

namespace Lab\_5\_6

{

public partial class Form1 : Form

{

bool lightButtonPressed = false;

bool pause;

float[] vievVertex;

float[] ribsViev;

float[] lightAmbient;

float[] lightPosition;

float[] lightDiffuse;

float[] lightSpecular;

float[] ambientMaterial;

float[] lightDirection;

float[,] color;

float[,] cubeVertexArray;

private bool nonNumberEntered;

OpenGL gl;

public Form1()

{

InitializeComponent();

vievVertex = new float[24];

ribsViev = new float[24];

color = new float[6, 3];

cubeVertexArray = new float[24, 3];

nonNumberEntered = false;

pause = false;

// Инициализируем точки куба.

initFigure();

initializeGL();

paintGL();

}

void paintGL()

{

// Когда источник света выключен.

if (lightButtonPressed == true)

{

lightAmbient = new float[] { 0.1f, 0.1f, 0.1f, 1f };

lightPosition = new float[] { 0.0f, 0.0f, 0.0f, 0.0f };

lightDiffuse = new float[] { 0.5f, 0.5f, 0.5f, 1.0f };

lightSpecular = new float[] { 0.8f, 0.8f, 0.8f, 1.0f };

ambientMaterial = new float[] { 0.2f, 0.2f, 0.2f, 1.0f };

lightDirection = new float[] { 0.0f, 0.0f, 0.0f, 0.0f };

}

// Когда источник света включен.

else

{

lightAmbient = new float[] { 0.1f, 0.1f, 0.1f, 1.0f };

lightPosition = new float[] { 1.0f, 1.0f, 0.0f, 1.0f };

lightDiffuse = new float[] { 0.5f, 0.5f, 0.5f, 1.0f };

lightSpecular = new float[] { 0.8f, 0.8f, 0.8f, 1.0f };

ambientMaterial = new float[] { 0.2f, 0.2f, 0.2f, 1.0f };

lightDirection = new float[] { 0.0f, 5.0f, 0.0f, 3.0f };

}

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

gl.LightModel(OpenGL.GL\_LIGHT\_MODEL\_AMBIENT, lightAmbient);

gl.Light(OpenGL.GL\_LIGHT0, OpenGL.GL\_POSITION, lightPosition);

gl.Light(OpenGL.GL\_LIGHT0, OpenGL.GL\_AMBIENT, lightAmbient);

gl.Light(OpenGL.GL\_LIGHT0, OpenGL.GL\_DIFFUSE, lightDiffuse);

gl.Light(OpenGL.GL\_LIGHT0, OpenGL.GL\_SPOT\_DIRECTION, lightDirection);

gl.Light(OpenGL.GL\_LIGHT0, OpenGL.GL\_SPECULAR, lightSpecular);

gl.Enable(OpenGL.GL\_LIGHTING);

gl.Enable(OpenGL.GL\_LIGHT0);

draw(); // Отрисовать сцену.

}

private void openGLControl1\_OpenGLDraw(object sender, SharpGL.RenderEventArgs args)

{

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

gl.LoadIdentity();

gl.Translate(-1.5f, 0.0f, -6.0f);

gl.Begin(OpenGL.GL\_QUADS);

draw();

}

void initFigure() // Точки куба.

{

// Верхняя грань.

color[0, 0] = 1;

color[0, 1] = 0;

color[0, 2] = 0;

cubeVertexArray[0, 0] = 1.0f;

cubeVertexArray[0, 1] = -1.0f;

cubeVertexArray[0, 2] = -1.0f;

cubeVertexArray[1, 0] = 1.0f;

cubeVertexArray[1, 1] = -1.0f;

cubeVertexArray[1, 2] = 1.0f;

cubeVertexArray[2, 0] = 1.0f;

cubeVertexArray[2, 1] = 1.0f;

cubeVertexArray[2, 2] = 1.0f;

cubeVertexArray[3, 0] = 1.0f;

cubeVertexArray[3, 1] = 1.0f;

cubeVertexArray[3, 2] = -1.0f;

// Нижняя грань.

color[1, 0] = 0;

color[1, 1] = 1;

color[1, 2] = 0;

cubeVertexArray[4, 0] = -1.0f;

cubeVertexArray[4, 1] = -1.0f;

cubeVertexArray[4, 2] = -1.0f;

cubeVertexArray[5, 0] = -1.0f;

cubeVertexArray[5, 1] = -1.0f;

cubeVertexArray[5, 2] = 1.0f;

cubeVertexArray[6, 0] = -1.0f;

cubeVertexArray[6, 1] = 1.0f;

cubeVertexArray[6, 2] = 1.0f;

cubeVertexArray[7, 0] = -1.0f;

cubeVertexArray[7, 1] = 1.0f;

cubeVertexArray[7, 2] = -1.0f;

// Задняя грань.

color[2, 0] = 0;

color[2, 1] = 0;

color[2, 2] = 1;

cubeVertexArray[8, 0] = -1.0f;

cubeVertexArray[8, 1] = -1.0f;

cubeVertexArray[8, 2] = -1.0f;

cubeVertexArray[9, 0] = 1.0f;

cubeVertexArray[9, 1] = -1.0f;

cubeVertexArray[9, 2] = -1.0f;

cubeVertexArray[10, 0] = 1.0f;

cubeVertexArray[10, 1] = -1.0f;

cubeVertexArray[10, 2] = 1.0f;

cubeVertexArray[11, 0] = -1.0f;

cubeVertexArray[11, 1] = -1.0f;

cubeVertexArray[11, 2] = 1.0f;

//Левая грань.

color[3, 0] = 1;

color[3, 1] = 1;

color[3, 2] = 0;

cubeVertexArray[12, 0] = 1.0f;

cubeVertexArray[12, 1] = 1.0f;

cubeVertexArray[12, 2] = -1.0f;

cubeVertexArray[13, 0] = 1.0f;

cubeVertexArray[13, 1] = -1.0f;

cubeVertexArray[13, 2] = -1.0f;

cubeVertexArray[14, 0] = -1.0f;

cubeVertexArray[14, 1] = -1.0f;

cubeVertexArray[14, 2] = -1.0f;

cubeVertexArray[15, 0] = -1.0f;

cubeVertexArray[15, 1] = 1.0f;

cubeVertexArray[15, 2] = -1.0f;

// Правая грань.

color[4, 0] = 1;

color[4, 1] = 0;

color[4, 2] = 1;

cubeVertexArray[16, 0] = -1.0f;

cubeVertexArray[16, 1] = -1.0f;

cubeVertexArray[16, 2] = 1.0f;

cubeVertexArray[17, 0] = -1.0f;

cubeVertexArray[17, 1] = 1.0f;

cubeVertexArray[17, 2] = 1.0f;

cubeVertexArray[18, 0] = 1.0f;

cubeVertexArray[18, 1] = 1.0f;

cubeVertexArray[18, 2] = 1.0f;

cubeVertexArray[19, 0] = 1.0f;

cubeVertexArray[19, 1] = -1.0f;

cubeVertexArray[19, 2] = 1.0f;

// Передняя грань.

color[5, 0] = 0;

color[5, 1] = 1;

color[5, 2] = 1;

cubeVertexArray[20, 0] = 1.0f;

cubeVertexArray[20, 1] = 1.0f;

cubeVertexArray[20, 2] = 1.0f;

cubeVertexArray[21, 0] = -1.0f;

cubeVertexArray[21, 1] = 1.0f;

cubeVertexArray[21, 2] = 1.0f;

cubeVertexArray[22, 0] = -1.0f;

cubeVertexArray[22, 1] = 1.0f;

cubeVertexArray[22, 2] = -1.0f;

cubeVertexArray[23, 0] = 1.0f;

cubeVertexArray[23, 1] = 1.0f;

cubeVertexArray[23, 2] = -1.0f;

}

private void openGLControl1\_KeyPress(object sender, KeyPressEventArgs e)

{

if (nonNumberEntered == true)

e.Handled = true;

}

// Функция отрисовки объектов.

void draw()

{

float minZ = 5;

if (!pause)

{

for (int i = 0; i < 24; ++i)

{

if (cubeVertexArray[i, 2] < minZ)

{

minZ = cubeVertexArray[i, 2];

}

}

for (int i = 0, vertex = 0; i < 24; i += 4, ++vertex)

{

if ((cubeVertexArray[i, 2] == minZ) || (cubeVertexArray[i + 1, 2] == minZ) || (cubeVertexArray[i + 2, 2] == minZ) || (cubeVertexArray[i + 3, 2] == minZ))

vievVertex[vertex] = 0;

else

vievVertex[vertex] = 1;

}

for (int i = 0, ribs = 0; i < 24; i += 2, ++ribs)

{

if ((cubeVertexArray[i, 2] == minZ) || (cubeVertexArray[i + 1, 2] == minZ))

ribsViev[ribs] = 0;

else

ribsViev[ribs] = 1;

}

}

gl.LineWidth(1);

// Отрисовка усеченного куба по массиву из функции initFigure.

gl.LineWidth(4);

for (int i = 0, vertex = 0; i < 24; i += 4, vertex++)

// Пробегаем по всем точкам линий по порядку.

{

if (vievVertex[vertex] == 1)

{

gl.Color(color[vertex, 0], color[vertex, 1], color[vertex, 2]);

gl.Begin(OpenGL.GL\_QUADS);

gl.Vertex(cubeVertexArray[i, 0], cubeVertexArray[i, 1], cubeVertexArray[i, 2]);

gl.Vertex(cubeVertexArray[i + 1, 0], cubeVertexArray[i + 1, 1], cubeVertexArray[i + 1, 2]);

gl.Vertex(cubeVertexArray[i + 2, 0], cubeVertexArray[i + 2, 1], cubeVertexArray[i + 2, 2]);

gl.Vertex(cubeVertexArray[i + 3, 0], cubeVertexArray[i + 3, 1], cubeVertexArray[i + 3, 2]);

gl.End();

}

}

gl.Color(0.0, 0.0, 0.0);

for (int i = 0, ribs = 0; i < 24; i += 2, ribs++)

{

if (ribsViev[ribs] == 1)

{

gl.Begin(OpenGL.GL\_LINES);

gl.Vertex(cubeVertexArray[i, 0], cubeVertexArray[i, 1], cubeVertexArray[i, 2]);

gl.Vertex(cubeVertexArray[i + 1, 0], cubeVertexArray[i + 1, 1], cubeVertexArray[i + 1, 2]);

gl.End();

}

}

gl.Color(1.0, 1.0, 1.0);

gl.Begin(OpenGL.GL\_QUADS);

gl.Vertex(6.0, 6.0, -2.0);

gl.Vertex(6.0, -6.0, -2.0);

gl.Vertex(-6.0, -6.0, -2.0);

gl.Vertex(-6.0, 6.0, -2.0);

gl.End();

}

void initializeGL()

{

gl = openGLControl1.OpenGL;

gl.ClearColor(0.0f, 0.0f, 0.0f, 0.0f);

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

gl.ShadeModel(OpenGL.GL\_SMOOTH);

gl.Enable(OpenGL.GL\_COLOR\_MATERIAL);

gl.Enable(OpenGL.GL\_LIGHTING);

gl.Enable(OpenGL.GL\_LIGHT0);

gl.LightModel(OpenGL.GL\_LIGHT\_MODEL\_LOCAL\_VIEWER, OpenGL.GL\_TRUE);

}

// Поворот куба.

void matrixMult(float[,] matrix, float[,] bMatrix)

{

float[,] product = new float[24, 3];

for (int i = 0; i< 24; i++)

for (int j = 0; j< 3; j++)

{

product[i, j] = 0;

for (int k = 0; k< 3; k++)

product[i, j] += matrix[i, k] \* bMatrix[k, j];

}

for (int i = 0; i< 24; ++i)

for (int j = 0; j< 3; ++j)

{

matrix[i, j] = product[i, j];

}

if (!lightButtonPressed)

{

float[] temp = { 0, 0, 0 };

for (int j = 0; j< 3; j++)

{

for (int k = 0; k< 3; k++)

temp[j] += bMatrix[j, k] \* lightPosition[k];

}

for (int i = 0; i< 3; ++i)

{

lightPosition[i] = temp[i];

temp[i] = 0;

}

for (int j = 0; j< 3; j++)

{

for (int k = 0; k < 3; k++)

temp[j] += bMatrix[j, k] \* lightDirection[k];

}

for (int i = 0; i< 3; ++i)

lightDirection[i] = temp[i];

}

}

// Поворот на градусы.

void reflectX(float angle)

{

float[,] mult = {

{ 1.0f, 0.0f, 0.0f },

{ 0.0f, (float)Math.Cos(angle), -(float)Math.Sin(angle) },

{ 0.0f, (float)Math.Sin(angle), (float)Math.Cos(angle) }

};

matrixMult(cubeVertexArray, mult);

}

void reflectY(float angle)

{

float[,] mult = {

{ (float)Math.Cos(angle), 0.0f, (float)Math.Sin(angle) },

{ 0.0f, 1.0f, 0.0f },

{ -(float)Math.Sin(angle), 0.0f, (float)Math.Cos(angle) }

};

matrixMult(cubeVertexArray, mult);

}

void reflectZ(float angle)

{

float[,] mult = {

{ (float)Math.Cos(angle), -(float)Math.Sin(angle), 0.0f, },

{ (float)Math.Sin(angle), (float)Math.Cos(angle), 0.0f },

{ 0.0f, 0.0f, 1.0f }

};

matrixMult(cubeVertexArray, mult);

}

private void openGLControl1\_KeyDown(object sender, KeyEventArgs e)

{

nonNumberEntered = false;

switch (e.KeyCode) {

case Keys.D: reflectY(-0.05f); break;

case Keys.A: reflectY(+0.05f); break;

case Keys.W: reflectX(+0.05f); break;

case Keys.S: reflectX(-0.05f); break;

case Keys.E: reflectZ(+0.05f); break;

case Keys.Q: reflectZ(-0.05f); break;

}

}

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

{

if (lightButtonPressed == false)

{

LightButton.Text = "Включить источник света";

lightButtonPressed = true;

}

else

{

LightButton.Text = "Отключить источник света";

lightButtonPressed = false;

}

paintGL();

}

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

{

if (pause == false)

{

LineButton.Text = "Возобновить выявление граней";

pause = true;

}

else

{

LineButton.Text = "Прервать выявление граней";

pause = false;

}

}}}

**Вывод**

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