**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра вычислительной техники**

**Отчет**

**по лабораторной работе №6**

**по дисциплине «Компьютерная графика»**

**Тема: "** **Формирования реалистических изображений с использованием простых моделей освещения одним или двумя точечными источниками ”**

|  |  |  |
| --- | --- | --- |
| Студент гр. 6307 |  | Лазарев С. О. |
| Преподаватель |  | Матвеева И. В. |

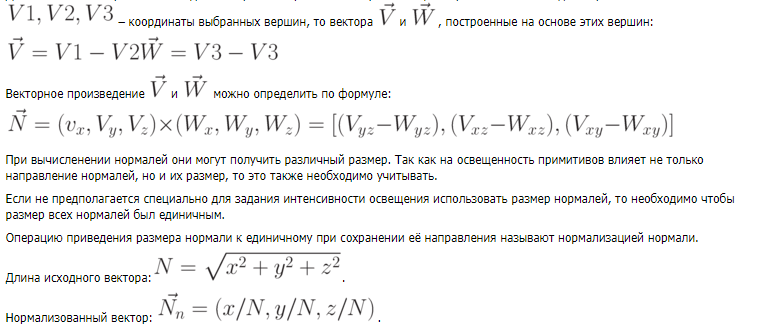
Санкт-Петербург

2019

**Задание:**

Сформировать для наблюдателя изображение освещенных многоугольников или поверхностей, сформированных при выполнении темы 5 без выявления теней. Известными параметрами являются цвет каждого многогранника, интенсивность излучения точечного источника, его положение и положение наблюдателя в пространстве. Обеспечить поворот окрашенной поверхности вокруг осей X и Y или изменение координат источников освещения и наблюдателя.

**Теоретическая часть:**



**Иллюстрация работы приложения:**

Приложение написано на языке Dart, фреймворк Flutter. Тестирование проводилось на устройстве iPhone X.

Повороты выполнены из состояния инициализации.

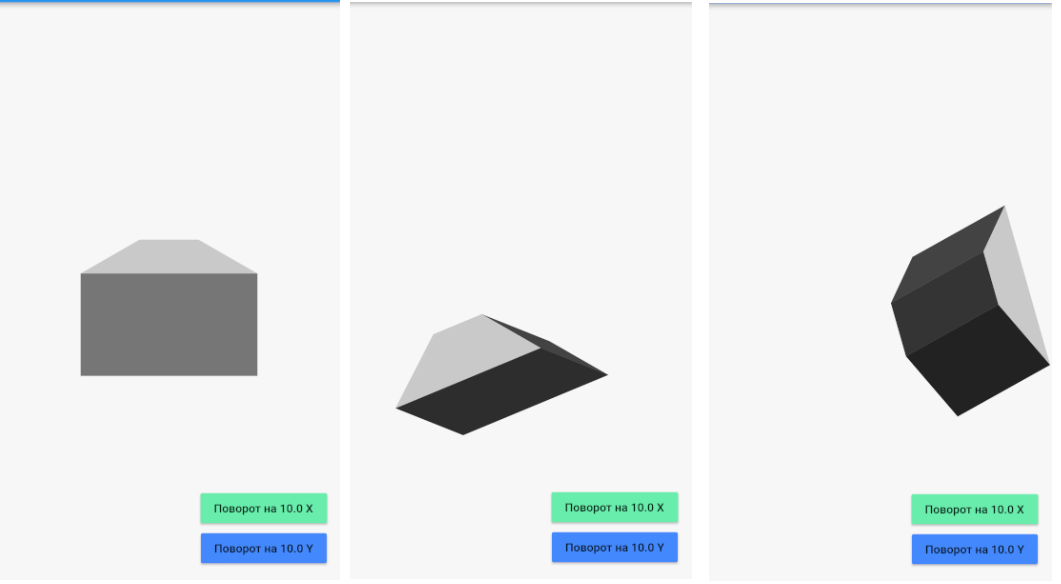


Рис. 1. Иллюстрация работы приложения

**Вывод:**

Я научился убирать невидимые ребра выпуклого тела, используя алгоритм Робертса. Реализовал поворот объемного тела относительно осей X и Y при помощи матриц поворота. Также научился освещать объекты. Получил навык работы с изометрической проекцией.

**Код:**

class MyPainter extends CustomPainter {

@override

void paint(Canvas canvas, Size size) {

// i

vector.Vector3 pt = toIsometric(p);

vector.Vector3 rt = toIsometric(r);

vector.Vector3 st = toIsometric(s);

vector.Vector3 tt = toIsometric(t);

vector.Vector3 ut = toIsometric(u);

vector.Vector3 vt = toIsometric(v);

vector.Vector3 yt = toIsometric(y);

vector.Vector3 qt = toIsometric(q);

var i1 = rt.x \* qt.y - qt.x \* rt.y;

var i2 = yt.x \* tt.y - tt.x \* yt.y;

var i3 = ut.x \* pt.y - pt.x \* ut.y;

var i4 = pt.x \* qt.y - pt.y \* qt.x;

var i5 = qt.x \* rt.y - rt.x \* qt.y;

var i6 = rt.x \* ut.y - ut.x \* rt.y;

// k

var k1 = ut.x \* rt.y - rt.x \* ut.y;

var k2 = vt.x \* yt.y - yt.x \* vt.y;

var k3 = yt.x \* ut.y - ut.x \* yt.y;

var k4 = tt.x \* pt.y - pt.x \* tt.y;

var k5 = st.x \* qt.y - qt.x \* st.y;

var k6 = vt.x \* rt.y - rt.x \* vt.y;

// j

var j1 = -(ut.x \* qt.y - qt.x \* ut.y);

var j2 = -(vt.x \* tt.y - tt.x \* vt.y);

var j3 = -(yt.x \* pt.y - pt.x \* yt.y);

var j4 = -(tt.x \* qt.y - qt.x \* tt.y);

var j5 = -(st.x \* rt.y - rt.x \* st.y);

var j6 = -(vt.x \* ut.y - ut.x \* vt.y);

double mporiPUQR = (i1 + k1 + j1);

double mporiTSVY = (i2 + k2 + j2);

double mporiPUYT = (i3 + k3 + j3);

double mporiTPQS = (i4 + k4 + j4);

double mporiQSVR = (i5 + k5 + j5);

double mporiURVY = (i6 + k6 + j6);

vector.Vector3 n1 = new vector.Vector3(i1, k1, j1);

vector.Vector3 n2 = new vector.Vector3(i2, k2, j2);

vector.Vector3 n3 = new vector.Vector3(i3, k3, j3);

vector.Vector3 n4 = new vector.Vector3(i4, k4, j4);

vector.Vector3 n5 = new vector.Vector3(i5, k5, j5);

vector.Vector3 n6 = new vector.Vector3(i6, k6, j6);

n1.normalize();

n2.normalize();

n3.normalize();

n4.normalize();

n5.normalize();

n6.normalize();

vector.Vector3 svett = toIsometric(svet);

var oriPUQR = sign(mporiPUQR);

var oriTSVY = sign(mporiTSVY);

var oriPUYT = sign(mporiPUYT);

var oriTPQS = sign(mporiTPQS);

var oriQSVR = sign(mporiQSVR);

var oriURVY = sign(mporiURVY);

var visPU = sqrt(sign(oriPUQR + oriPUYT + 1)); //

var visPQ = sqrt(sign(oriTPQS + oriPUQR + 1)); //

var visQR = sqrt(sign(oriQSVR + oriPUQR + 1)); //

var visRU = sqrt(sign(oriPUQR + oriURVY + 1)); //

var visYV = sqrt(sign(oriTSVY + oriURVY + 1)); //

var visYT = sqrt(sign(oriTSVY + oriPUYT + 1)); //

var visTS = sqrt(sign(oriTSVY + oriTPQS + 1)); //

var visSV = sqrt(sign(oriTSVY + oriQSVR + 1)); //

var visPT = sqrt(sign(oriPUYT + oriTPQS + 1));

var visUY = sqrt(sign(oriPUYT + oriURVY + 1));

var visRV = sqrt(sign(oriQSVR + oriURVY + 1));

var visQS = sqrt(sign(oriQSVR + oriTPQS + 1));

var pathPT = new Path();

var pathUY = new Path();

var pathRV = new Path();

var pathQS = new Path();

var pathTS = new Path();

var pathSV = new Path();

var paintPT = new Paint()

..color = Color.fromARGB(

255 \* num1.toInt(), 255 \* num1.toInt(), 255 \* num1.toInt(), 255)

..style = PaintingStyle.fill

..strokeWidth = 4.0;

var paintUY = new Paint()

..color = Color.fromARGB(

255 \* num2.toInt(), 255 \* num2.toInt(), 255 \* num2.toInt(), 255)

..style = PaintingStyle.fill

..strokeWidth = 4.0;

var paintRV = new Paint()

..color = Color.fromARGB(

255 \* num3.toInt(), 255 \* num3.toInt(), 255 \* num3.toInt(), 255)

..style = PaintingStyle.fill

..strokeWidth = 4.0;

var paintQS = new Paint()

..color = Color.fromARGB(

255 \* num4.toInt(), 255 \* num4.toInt(), 255 \* num4.toInt(), 255)

..style = PaintingStyle.fill

..strokeWidth = 4.0;

var paintTS = new Paint()

..color = Color.fromARGB(

255 \* num5.toInt(), 255 \* num5.toInt(), 255 \* num5.toInt(), 255)

..style = PaintingStyle.fill

..strokeWidth = 4.0;

var paintSV = new Paint()

..color = Color.fromARGB(

255 \* num6.toInt(), 255 \* num6.toInt(), 255 \* num6.toInt(), 255)

..style = PaintingStyle.fill

..strokeWidth = 4.0;

/\*

if (!visPT.isNaN) {

//

pathPT.moveTo(pt.x + center.dx, -pt.y + center.dy);

pathPT.lineTo(tt.x + center.dx, -tt.y + center.dy);

canvas.drawPath(pathPT, paintPT);

}

if (!visUY.isNaN) {

//

pathUY.moveTo(ut.x + center.dx, -ut.y + center.dy);

pathUY.lineTo(yt.x + center.dx, -yt.y + center.dy);

canvas.drawPath(pathUY, paintUY);

}

if (!visRV.isNaN) {

//

pathRV.moveTo(rt.x + center.dx, -rt.y + center.dy);

pathRV.lineTo(vt.x + center.dx, -vt.y + center.dy);

canvas.drawPath(pathRV, paintRV);

}

if (!visQS.isNaN) {

//

pathQS.moveTo(qt.x + center.dx, -qt.y + center.dy);

pathQS.lineTo(st.x + center.dx, -st.y + center.dy);

canvas.drawPath(pathQS, paintQS);

}

if (!visRU.isNaN) {

//

pathRU.moveTo(rt.x + center.dx, -rt.y + center.dy);

pathRU.lineTo(ut.x + center.dx, -ut.y + center.dy);

canvas.drawPath(pathRU, paintRU);

}

}

}