**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра вычислительной техники**

**Отчет**

**по лабораторной работе №5**

**по дисциплине «Компьютерная графика»**

**Тема: "** **Исследование алгоритмов выявления видимости сложных сцен”**

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

Санкт-Петербург

2019

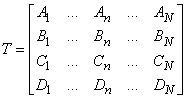
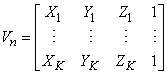
**Задание:**

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

**Теоретическая часть:**

Алгоритм получения изображений одиночных выпуклых объектов, составленных из плоских граней, был впервые разработан Робертсом. Алгоритм Робертса может быть с успехом применен для изображения множества выпуклых многогранников на одной сцене в виде проволочной модели с удаленными невидимыми линиями. Метод не пригоден непосредственно для передачи падающих теней и других сложных визуальных эффектов. Рассмотрим основные идеи метода на примере одиночного выпуклого многогранника.

Объект - [многогранник](http://sernam.ru/book_e_math.php?id=78), состоящий из http://www.sernam.ru/archive/arch.php?path=../htm/book_graph3d/files.book&file=graph3d_44.files/image001.gif граней, описывается матрицей http://www.sernam.ru/archive/arch.php?path=../htm/book_graph3d/files.book&file=graph3d_44.files/image002.gif размера http://www.sernam.ru/archive/arch.php?path=../htm/book_graph3d/files.book&file=graph3d_44.files/image003.gif, каждый http://www.sernam.ru/archive/arch.php?path=../htm/book_graph3d/files.book&file=graph3d_44.files/image004.gif-й столбец которой содержит описание http://www.sernam.ru/archive/arch.php?path=../htm/book_graph3d/files.book&file=graph3d_44.files/image004.gif-й http://www.sernam.ru/archive/arch.php?path=../htm/book_graph3d/files.book&file=graph3d_44.files/image005.gif плоскости http://www.sernam.ru/archive/arch.php?path=../htm/book_graph3d/files.book&file=graph3d_44.files/image006.gif в объектной системе координат http://www.sernam.ru/archive/arch.php?path=../htm/book_graph3d/files.book&file=graph3d_44.files/image007.gif:

. 

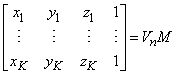
Минимальное число граней http://www.sernam.ru/archive/arch.php?path=../htm/book_graph3d/files.book&file=graph3d_44.files/image009.gif наблюдается у тетраэдра.

Важное требование к модели объекта в алгоритме Робертса заключается в достижении такой формы (знака) в функции каждой плоскости http://www.sernam.ru/archive/arch.php?path=../htm/book_graph3d/files.book&file=graph3d_44.files/image010.gif, при которой справедливо http://www.sernam.ru/archive/arch.php?path=../htm/book_graph3d/files.book&file=graph3d_44.files/image011.gif для любой точки http://www.sernam.ru/archive/arch.php?path=../htm/book_graph3d/files.book&file=graph3d_44.files/image012.gif, заведомо принадлежащей телу многогранника. Достижение этого условия осуществляется путем опытной проверки знака функции относительно внутренней точки, в качестве которой может выступать точка со средним геометрическим положением относительно всех вершин многогранника. Формально правило формирования "положительной" грани может быть представлено в виде http://www.sernam.ru/archive/arch.php?path=../htm/book_graph3d/files.book&file=graph3d_44.files/image013.gif, где http://www.sernam.ru/archive/arch.php?path=../htm/book_graph3d/files.book&file=graph3d_44.files/image014.gif - функция, описывающая http://www.sernam.ru/archive/arch.php?path=../htm/book_graph3d/files.book&file=graph3d_44.files/image004.gif-ю грань с априори неизвестным знаком относительно внутренней точки; http://www.sernam.ru/archive/arch.php?path=../htm/book_graph3d/files.book&file=graph3d_44.files/image015.gif - та же функция, но с положительным значением. При достижении "положительности" всех граней автоматически достигается ориентация [нормали](http://sernam.ru/book_e_math.php?id=89) http://www.sernam.ru/archive/arch.php?path=../htm/book_graph3d/files.book&file=graph3d_44.files/image016.gif к любой из граней внутрь фигуры.

Кроме параметров уравнений граней необходимо знать координаты вершин каждой грани. Вершины могут быть заданы или вычислены из матрицы http://www.sernam.ru/archive/arch.php?path=../htm/book_graph3d/files.book&file=graph3d_44.files/image002.gif путем определения общих решений каждой плоскости со всеми остальными. В любом случае для каждой http://www.sernam.ru/archive/arch.php?path=../htm/book_graph3d/files.book&file=graph3d_44.files/image004.gif-й грани должна быть составлена [матрица](http://sernam.ru/lect_math1.php?id=5) http://www.sernam.ru/archive/arch.php?path=../htm/book_graph3d/files.book&file=graph3d_44.files/image017.gif - числа вершин.

He все грани, входящие в состав объекта, будут видны наблюдателю. Невидимой будет та, [нормаль](http://sernam.ru/book_e_math.php?id=89) http://www.sernam.ru/archive/arch.php?path=../htm/book_graph3d/files.book&file=graph3d_45.files/image001.gif к которой ориентирована в ту же сторону, что и [вектор](http://sernam.ru/lect_math1.php?id=14) http://www.sernam.ru/archive/arch.php?path=../htm/book_graph3d/files.book&file=graph3d_45.files/image002.gif, направленный от объекта в сторону наблюдателя, т. е. при выполнении условия неотрицательности скалярного произведения http://www.sernam.ru/archive/arch.php?path=../htm/book_graph3d/files.book&file=graph3d_45.files/image003.gif.

Эти грани исключаются из дальнейшего анализа. Все оставшиеся являются видимыми. Если грань http://www.sernam.ru/archive/arch.php?path=../htm/book_graph3d/files.book&file=graph3d_45.files/image004.gif - видимая, а объектная http://www.sernam.ru/archive/arch.php?path=../htm/book_graph3d/files.book&file=graph3d_45.files/image005.gif и экранная http://www.sernam.ru/archive/arch.php?path=../htm/book_graph3d/files.book&file=graph3d_45.files/image006.gif системы координат связаны преобразованием http://www.sernam.ru/archive/arch.php?path=../htm/book_graph3d/files.book&file=graph3d_45.files/image007.gif (3.2.1), т.е. http://www.sernam.ru/archive/arch.php?path=../htm/book_graph3d/files.book&file=graph3d_45.files/image008.gif, то для ортогональной проекции координаты вершин http://www.sernam.ru/archive/arch.php?path=../htm/book_graph3d/files.book&file=graph3d_45.files/image009.gif многоугольника-проекции на изображении вычисляются следующим образом:

.

Область внутри такой фигуры закрашивается соответствующим значением интенсивности, для вычисления которой и определения факта затененности телом самого многогранника вычисляют [скалярное произведение](http://stu.sernam.ru/lect_alg.php?id=174) нормального вектора http://www.sernam.ru/archive/arch.php?path=../htm/book_graph3d/files.book&file=graph3d_45.files/image001.gif и вектора http://www.sernam.ru/archive/arch.php?path=../htm/book_graph3d/files.book&file=graph3d_45.files/image011.gif, нацеленного из объекта на источник света. Грань освещена при отрицательности скалярного произведения, а величина освещенности выбирается пропорциональной значению http://www.sernam.ru/archive/arch.php?path=../htm/book_graph3d/files.book&file=graph3d_45.files/image012.gif. Соответственно грань находится в тени при условии http://www.sernam.ru/archive/arch.php?path=../htm/book_graph3d/files.book&file=graph3d_45.files/image013.gif. Таким образом, алгоритм Робертса очень простыми действиями достигает успешного построения изображений отдельных выпуклых многогранных объектов.

Основным недостатком метода, определившим ограниченность его распространения, являются неспособность без привлечения других подходов реализовать падающие тени, невозможность передачи зеркальных эффектов и [преломления света](http://sernam.ru/book_phis_t3.php?id=82) и, наконец, строгая ориентация метода только на [выпуклые многогранники](http://edu.sernam.ru/book_kiber1.php?id=265).

**Иллюстрация работы приложения:**

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

Повороты выполнены из состояния инициализации.

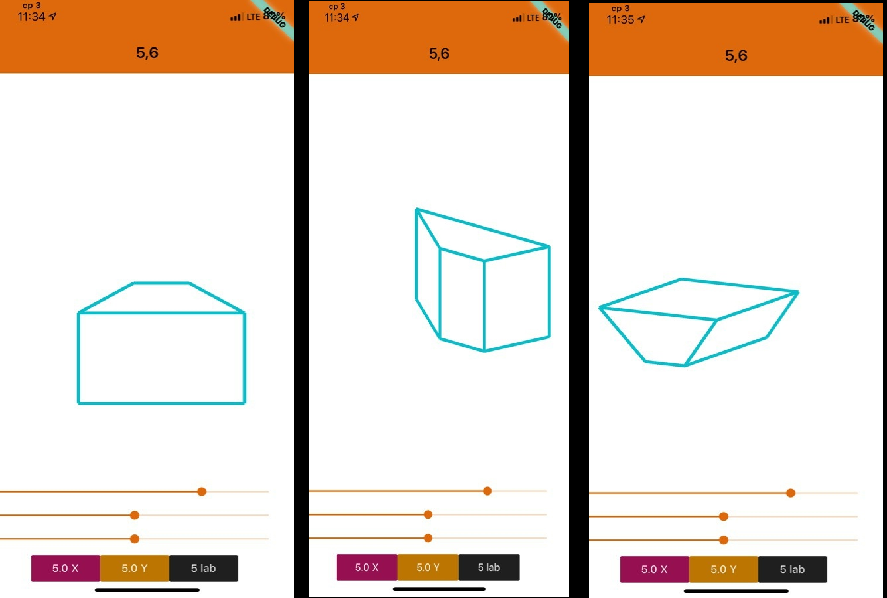


Рис. 1. Иллюстрация работы приложения/

**Вывод:**

Я научился убирать невидимые ребра выпуклого тела, используя алгоритм Робертса. Реализовал поворот объемного тела относительно осей X и Y при помощи матриц поворота. 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);

}

}

}

@override

bool shouldRepaint(MyPainter oldDelegate) => true;

}