**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра вычислительной техники**

**отчет**

**по лабораторной работе №3**

**по дисциплине «Компьютерная графика»**

**Тема: "** **Формирования различных поверхностей с использованием ее пространственного разворота и ортогонального проецирования на плоскость при ее визуализации”**

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

Санкт-Петербург

2019

**Задание:**

Сформировать билинейную поверхность на основе произвольного задания ее четырех угловых точек. Обеспечить ее поворот относительно осей X и Y.

**Теоретическая часть:**

Билинейная поверхность конструируется из четырех угловых точек единичного квадрата в параметрическом пространстве, т.е. из точек

P(0,0), P(0,1) , P(1,0), P(1,1).

Любая точка на поверхности определяется линейной интерполяцией между противоположными границами единичного квадрата, как это показано на рис.1.

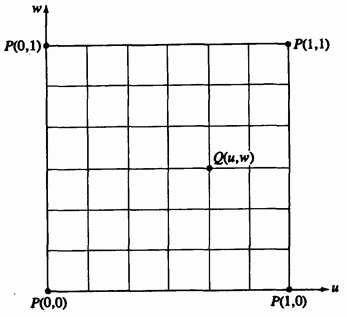


Рис. 1. Билинейная интерполяция в параметрическом пространстве.

Любая точка внутри параметрического квадрата задается уравнением:

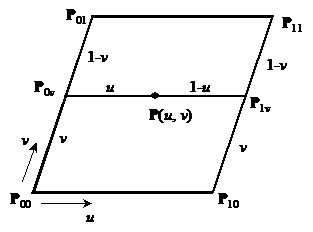


Рис. 2. Билинейная поверхность

http://www.sernam.ru/archive/arch.php?path=../htm/book_mm3d/files.book&file=mm3d_105.files/image005.gif,

0<=u<=1 и 0<=v<=1

Если координатные векторы четырех точек, определяющих билинейную поверхность, заданы в трехмерном объектном пространстве, то будет трехмерна и билинейная поверхность, получаемая в результате отображения параметрического пространства в объектное. Если четыре определяющие точки не лежат в одной плоскости, то и билинейная поверхность также не лежит ни в какой плоскости. Действительно, в общем случае она сильно изогнута, пример этого показан на рис. 2. Определяющие точки являются концами противоположных диагоналей на противоположных гранях единичного куба. В результате получаем гиперболический параболоид

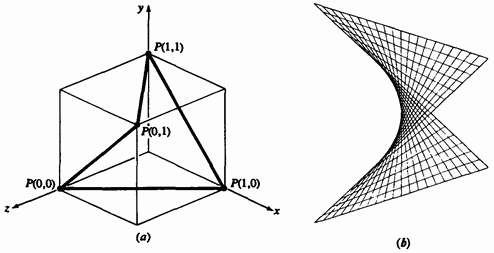


Рис. 3. Билинейная поверхность. (а) Определяющие угловые точки; (б) поверхность

**Иллюстрация работы приложения:**

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

Повороты выполнены из состояния инициализации.

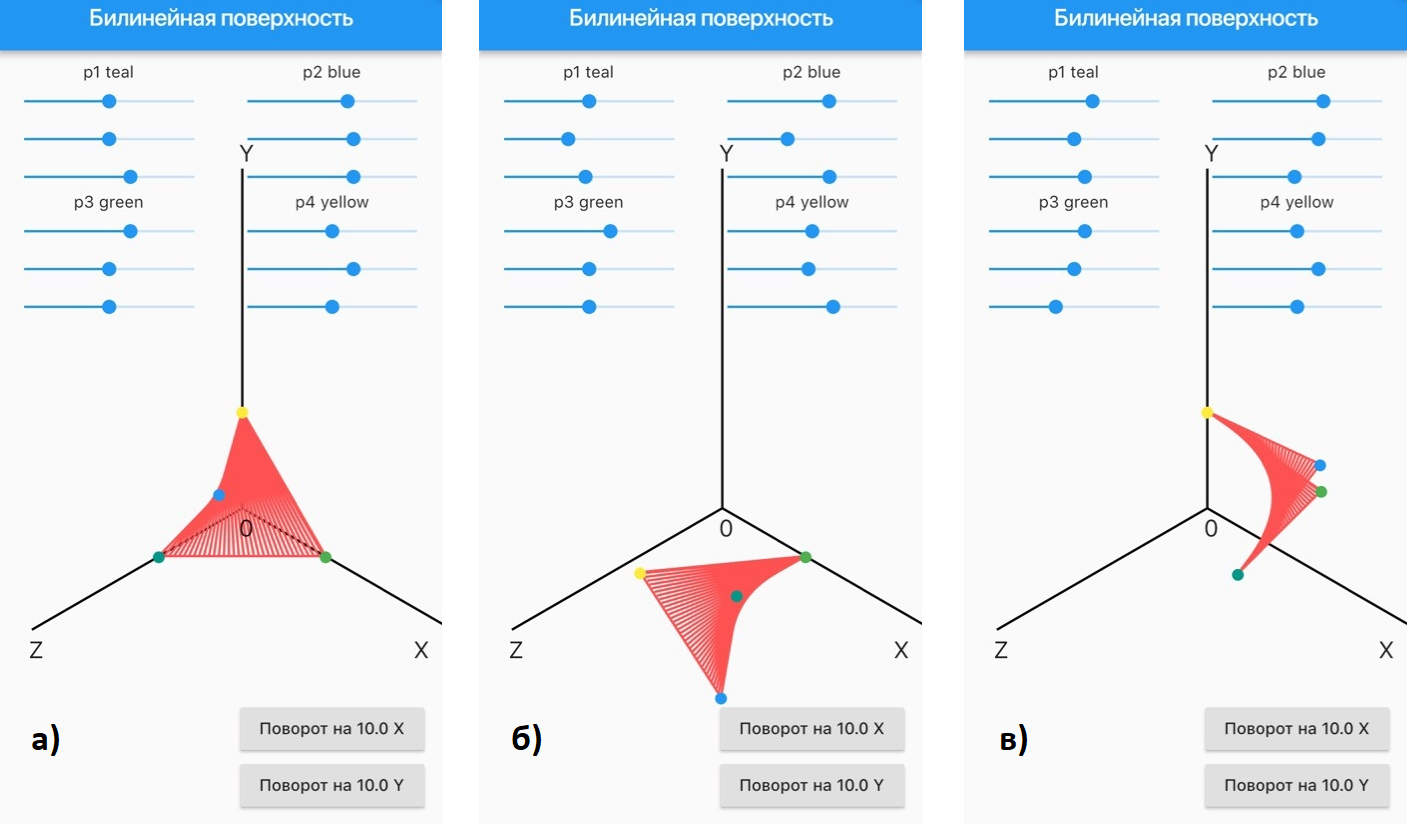


Рис. 4. Иллюстрация работы приложения. (а) Инициализация; (б) Поворот относительно оси Х на 100 градусов; (в) Поворот относительно оси Y на 60 градусов.

**Вывод:**

Я научился строить билинейную поверхность, используя линейную интерполяцию. Реализовал поворот поверхности относительно осей X и Y при помощи матриц поворота. Получил навык работы с изометрической проекцией.

**Код:**

surface(v.Vector3 p1, v.Vector3 p2, v.Vector3 p3, v.Vector3 p4) {

List<v.Vector3> list = new List<v.Vector3>();

double x = 0;

double y = 0;

double z = 0;

for (double uc = 0; uc <= 1; uc = uc + 0.03) {

for (double vc = 0; vc <= 1; vc = vc + 0.03) {

x = p1.x \* (1 - uc) \* (1 - vc) +

p2.x \* (1 - uc) \* vc +

p3.x \* (1 - vc) \* uc +

p4.x \* uc \* vc;

y = p1.y \* (1 - uc) \* (1 - vc) +

p2.y \* (1 - uc) \* vc +

p3.y \* (1 - vc) \* uc +

p4.y \* uc \* vc;

z = p1.z \* (1 - uc) \* (1 - vc) +

p2.z \* (1 - uc) \* vc +

p3.z \* (1 - vc) \* uc +

p4.z \* uc \* vc;

list.add(new v.Vector3(x, y, z));

}

}

return list;

}

class ShapesPainter extends CustomPainter {

@override

void paint(Canvas canvas, Size size) {

var path = new Path();

// AXIS

path.moveTo(center.x, center.y);

path.lineTo(axisX.x, axisX.y);

path.moveTo(center.x, center.y);

path.lineTo(axisZ.x, axisZ.y);

path.moveTo(center.x, center.y);

path.lineTo(center.x, 100);

canvas.drawPath(

path,

Paint()

..color = Colors.black

..style = PaintingStyle.stroke

..strokeWidth = 2.0,

);

path.close();

// AXIS

listDots = surface(p1, p2, p3, p4);

// Lines

var linePath = new Path();

linePath.moveTo(toIsometric(p1).x, toIsometric(p1).y);

linePath.lineTo(toIsometric(p3).x, toIsometric(p3).y);

linePath.lineTo(toIsometric(p4).x, toIsometric(p4).y);

linePath.lineTo(toIsometric(p2).x, toIsometric(p2).y);

linePath.lineTo(toIsometric(p1).x, toIsometric(p1).y);

linePath.moveTo(toIsometric(listDots[0]).x, toIsometric(listDots[0]).y);

for (int i = 0; i < listDots.length - 1; i++) {

linePath.lineTo(toIsometric(listDots[i]).x, toIsometric(listDots[i]).y);

linePath.lineTo(

toIsometric(listDots[i + 1]).x, toIsometric(listDots[i + 1]).y);

}

canvas.drawPath(

linePath,

Paint()

..color = Colors.redAccent

..style = PaintingStyle.stroke

..strokeWidth = 2.0,

);

linePath.close();

// Lines

}

@override

bool shouldRepaint(CustomPainter oldDelegate) => true;

}

drawPoint(toIsometric(p1), Colors.teal),

drawPoint(toIsometric(p2), Colors.blue),

drawPoint(toIsometric(p3), Colors.green),

drawPoint(toIsometric(p4), Colors.yellow),

drawAxisMarker('0', v.Vector3(center.x, center.y, 0)),

drawAxisMarker('X', v.Vector3(axisX.x, axisX.y, 0)),

drawAxisMarker('Y', v.Vector3(axisY.x, axisY.y - 30, 0)),

drawAxisMarker('Z', v.Vector3(axisZ.x, axisZ.y, 0)),