**Библиотека функций WebGeometry для работы с 2D и 3D объектами**

**function Vector2D(x, y)**

Функция-конструктор двумерных векторов Vector2D

**Пример**: Создание двумерного вектора с координатами (2, 1).

var vec = new Vector2D(2, 1); // vec = [2, 1]

**Пример**: Создание (объявление) пустого двумерного вектора.

Координаты вектора x и y - пока неопределены.

var vec = new Vector2D(); // vec = [undefined, undefined]

**Vector2D.prototype.Add = function(vector)**

Сложение двумерных векторов. Функция возвращает новый вектор, получившийся в результат сложения данного вектора this с вестором vector.

**Пример:**

var vec = new Vector2D(2, 1);

var vecOut = vec.Add(new Vector2D(3, 2)); // vecOut = [5, 3]

**Vector2D.prototype.Subtract = function(vector)**

Вычитание из данного вектора this вектора vector. Функция возвращает новый вектор, получившийся в результат вычитания.

**Пример:**

var vec = new Vector2D(2, 1);

var vecOut = vec.Subtract(new Vector2D(3, 2)); // vecOut = [-1, -1]

**Vector2D.prototype.Dot = function(vector)**

Скалярное произведение векторов this и vector. Функция возвращает результат скалярного произведения векторов.

**Пример:**

var vec1 = new Vector2D(2, 1);

var vec2 = new Vector2D(4, 3);

var dot = vec1.Dot(vec2); // dot = 11

**Vector2D.prototype.Normer = function()**

Нормировка данного вектора this.

**Пример:**

vec1 = new Vector2D(1, 1);

vec1.Normer(); // vec1 = [0.7071067811865475, 0.7071067811865475]

vec2 = new Vector2D(3, 1);

vec2.Normer(); // vec2 = [0.9486832980505138, 0.31622776601683794]

**Vector2D.prototype.Rotate = function(angle)**

Вращение на плоскости вектора this на угол angle. Функция возвращает новый вектор, получившийся в результат поворота.

**Пример:** Вращение вектора vec на 90 градусов по часовой стрелке.

var radian = 90 \* Math.PI / 180;

vec = new Vector2D(1, 0);

var vecOut = vec.Rotate(radian); // vecOut = [0, -1]

**Vector2D.prototype.Translate = function(vector**)

Перенос вектора this на вектор vector. Фактически this = this + vector

**function Matrix2D(m00, m01, m10, m11)**

Функция-конструктор двумерных матриц Matrix2D

[m00, m01]

[m10, m11]

**Пример**: Создание 2D-матрицы.

var mat = new Matrix2D(0, 1, 2, 3); // mat = [0, 1]

// [2, 3]

**Matrix2D.prototype.Det = function()**

Функция возвращает величину определителя матрицы

**Пример**: Вычисление детермината 2D-матрицы.

var mat = new Matrix2D(0, 1, 2, 3);

var det = mat.Det(); // det = -2

**Matrix2D.prototype.Inverse = function();**

Инверсия исходной матрицы this. Функция возвращает обратную матрицу. Если детерминат матрицы this равен 0, то функция возвращает null.

**Пример:**  Вычисление обратной матрицы.

var mat1 = new Matrix2D(0, 1, 2, 3);

var inv1 = mat1.Inverse(); // inv1 = [0, 0.5]   
 [1, -1.5]

var mat2 = new Matrix2D(1, 1, 1, 1); // det = 0

var inv2 = mat2.Inverse(); // inv2 = null

**Matrix2D.prototype.Transpose = function();**

Транспонирование матрицы.Функция возвращает транспонированную матрицу this.

**Пример:**

var mat = new Matrix2D(0, 1, 2, 3); // mat = [0, 1]

// [2, 3]

var tr = mat.transpose(); // tr = [0, 2]

// [1, 3]

**Matrix2D.prototype.** **MultiplyMatrixVector = function(vector);**

Умножение матрицы на вектор. Функция возвращает вектор, получившийся в результате умножения матрицы this на вектор vector.

**Пример:**

var vec = new Vector2D(1, 3);

var mat = new Matrix2D(1, 2, 3, 4);

var vecOut = mat. MultiplyMatrixVector (vec); // vecOut = [7, 15]

**function Point2D(x, y);**

Функция-конструктор точек на плоскости Point2D.

**Пример**: Создание точки с координатами (2, 1).

var point = new Point2D(2, 1);

**Пример**:

Создание(объявление) точки, кординаты которой пока не определены. Координаты точки, если потребуется, будут определены в программе в дальнейшем.

var point = new Point2D(); // point = [undefined, undefined]

**Point2D.prototype.Add = function(point);**

Сложение координат двух точек на плоскости. Функция возвращает точку с координатами равными сумме координт данной точки this и точки point.

**Пример:**

var pt1 = new Point2D(1, 2);

var pt2 = new Point2D(3, 4);

var ptOut = pt1.Add(pt2); // ptOut = [4, 6]

**Point2D.prototype.Translate = function(vector);**

Перенос точки на величину, заданную вектором. Функция возвращает точку с координатами точки this смещенными на значение вектора vector.

**Пример:**

var pt = new Point2D(1, 2);

var vec = new Vector2D(3, 4);

var ptOut = pt.Translate(vec); // ptOut = [4, 6]

**Point2D.prototype.Distance = function(point)**

Функция возвращает значение расстояния между точками this и point.

**Пример:**

var pt1 = new Point2D(1, 1);

var pt2 = new Point2D(2, 2);

var dist = pt1.Distance(pt2); // dist = 1.4142135623730951

**function Line2D(point1, point2)**

Функция-конструктор прямой Line2D. Прямая задается точками point1 и point2.

**Пример:**

var line = new Line2D(); // directCos = [undefined, undefined]

// distOXY = undefined

var line1 = new Line2D(new Point2D(-1, 0), new Point2D(0, 1));

// directCos = [0.7071067811865475, -0.7071067811865475]

// distOXY = -0.7071067811865475

var line2 = new Line2D(new Point2D(1, 0), new Point2D(0, 1));

// directCos = [0.7071067811865475, 0.7071067811865475]

// distOXY = 0.7071067811865475

**Line2D.prototype.NormalLine = function()**

Функция создает прямую на плоскости, перпендикулярную к исходной прямой.

**Пример:**

var line = new Line2D(new Point2D(-1, 0), new Point2D(0, 1));

// directCos = [0.7071067811865475, -0.7071067811865475]

// distOXY = -0.7071067811865475

var lineNormal = line.NormalLine();

// directCos = [0.7071067811865475, 0.7071067811865475]

// distOXY = -0.7071067811865475

**Line2D.prototype.CreateNormalLinePoint = function(point)**

Функция создает прямую на плоскости перпендикулярную к прямой this и проходящую через точку point.

**Пример:**

var line = new Line2D(new Point2D(-1, 0), new Point2D(0, 1));

// directCos = [0.7071067811865475, -0.7071067811865475]

// distOXY = -0.7071067811865475

var point = new Point2D(0, 1); // point = [0, 1]

var lineNormal = line.CreateNormalLinePoint(point);

// directCos = [0.7071067811865475, 0.7071067811865475]

// distOXY = 0.7071067811865475

**Line2D.prototype.IntersectionTwoLines = function(line)**

Функция возвращает точку лежащую на пересечении двух прямых на плоскости this и line.

**Пример:**

var line1 = new Line2D(new Point2D(-1, 0), new Point2D( 1, 2));

var line2 = new Line2D(new Point2D( 1, 0), new Point2D(-1, 2));

var point = line1.IntersectionTwoLines(line2, point);

// point = [0, 1]

**Line2D.prototype.Angle = function(line)**

Функция возвращает значение угла на плоскости между прямыми this и line.

**Пример:**

var line1 = new Line2D(new Point2D(0, 0), new Point2D(1, 0));

var line2 = new Line2D(new Point2D(-1, 0), new Point2D(0, 1));

var line3 = new Line2D(new Point2D(1, 0), new Point2D(0, 1));

var degreeAng12 = (180 / Math.PI) \* line1.Angle(line2);

// degreeAng12 = 45°

var degreeAng13 = (180 / Math.PI) \* line1.Angle(line3);

// degreeAng13 = 135°

**Line2D.prototype.CreateLineVectorPoint = function(vector, point)**

Данная прямая на плоскости this создается по вектору направления vector и точке point лежащей на создаваемой прямой.

**Пример:**

var vector = new Vector2D(2, 1);

var point = new Point2D(3, 0);

var line = new Line2D(); // объявление прямой line

// назначение прямой line требуемых параметров

line.CreateLineVectorPoint(vector, point);

// directCos = [0.4472135954999579, -0.8944271909999159]

// distOXY = 1.3416407864998738

**Line2D.prototype.Distance = function(point)**

Функция возвращает значение расстояния на плоскости точки point от прямой this.

**Пример:**

var line = new Line2D(new Point2D(0, 0), new Point2D(0, 1));

var point = new Point2D(4, 2);

var dist = line.Distance(point); // dist = 4

**function Circle2D(point, radius)**

Функция-конструктор окружности на плоскости Circle2D. Окружность задается своим центром point и величиной радиуса radius.

**Пример:**

var cir = new Circle2D(new Point2D(4, 3), 2);

**Circle2D.prototype.IntersectionTwoCircles = function(circle, point1, point2)**

Функция определяет точки пересечения point1 и point2 двух окружностей. Первая окружность задается this, а вторая – параметром circle.

**Пример:**

var cir1 = new Circle2D(new Point2D(0, 0), 1);

var cir2 = new Circle2D(new Point2D(1, 0), 1);

var point1 = new Point2D();

var point2 = new Point2D();

if(!cir1.IntersectionTwoCircles(cir2, point1, point2))

return null;

// point1 = [0.5, 0.8660254037844386]

// point2 = [0.5, -0.8660254037844386]

**Circle2D.prototype.IntersectionLineCircle = function(line, point1, point2)**

Функция определяет точки пересечения на плоскости point1 и point2 окружности this с прямой line.

**Пример:**

var cir = new Circle2D(new Point2D(0, 0), 2);

var line = new Line2D(new Point2D(0,0), new Point2D(1,1));

var point1 = new Point2D();

var point2 = new Point2D();

if(!cir.IntersectionLineCircle (line, point1, point2))

return null;

// point1 = [ 1.4142135623730951, 1.4142135623730951]

// point2 = [-1.4142135623730951, -1.4142135623730951]

**function QuadraticEquation(a, b, c, rez)**

Решение квадратного уравнения. a, b, c - коэффициенты и свободный член уравнения. rez - результат решения (массив из двух значений).

**Пример:**

var a = 1;

var b = -4;

var c = 3;

var rez = [];

if(!QuadraticEquation(a, b, c, rez))

{

return null;

}

// rez = [3, 1]

**function Vector3D(x, y, z)**

Функция-конструктор векторов в пространстве Vector3D

**Пример:**

var vec1 = new Vector3D(3, 2, 1); // vec1 = [3, 2, 1]

var vec2 = new Vector3D(); // vec2 = [undefined, undefined, undefined]

**Vector3D.prototype.Add = function(vector)**

Сложение 3D векторов. Функция возвращает новый вектор, получившийся в результате сложения сложения векторов this и vector.

**Пример:**

var vec = new Vector3D(3, 2, 1);

var vecOut = vec.Add(new Vector3D(0, 1, 2)); // vecOut = [3, 3, 3]

**Vector3D.prototype.Subtract = function(vector)**

Вычитание из вектора this вектора vector. Функция возвращает новый вектор, получившийся в результат вычитания.

**Пример:**

var vec = new Vector3D(2, 1, 0);

var vecOut = vec.Subtract(new Vector3D(1, 0, 1));

// vecOut = [1, 1, -1]

**Vector3D.prototype.Dot = function(vector)**

Скалярное произведение векторов this и vector. Функция возвращает результат скалярного произведения.

**Пример:**

var vec1 = new Vector3D(2, 1, 1);

var vec2 = new Vector3D(4, 3, 2);

var dot = vec1.Dot(vec2); // dot = 13

**Vector3D.prototype.Normer = function()**

Нормировка вектора this.

**Пример:**

vec1 = new Vector3D(1, 1, 1);

vec1.Normer(); // vec1 = [0.577.., 0.577.., 0.577..]

vec2 = new Vector3D(3, 1, 1);

vec2.Normer(); // vec2 = [0.0.904.., 0.301.., 0.301..]

**Vector3D.prototype.Angle = function(vector)**

Возвращает значение угла между векторами this и vector.

**Пример:**

vec1 = new Vector3D(1, 1, 0);

vec2 = new Vector3D(1, 1, 1);

var degreeAng12 = (180 / Math.PI) \* vec1.Angle(vec2);

// degreeAng12 = 35.26438968275466°

**Vector3D.prototype.Cross = function(vector)**

Возвращает вектор, являющийся векторным произведением векторов this и vector.

**Пример:**

vec1 = new Vector3D(1, 0, 0);

vec2 = new Vector3D(0, 1, 0);

vecOut = vec1.Cross(vec2); // vecOut = [0, 0, 1]

**Vector3D.prototype.Rotate = function(rot)**

Функция возвращает вектор this повернутый на величину определяемую матрицей rot.

**Пример:**

vec = new Vector3D(1, 0, 0);

var teta = Math.PI/2;

var mat = new Matrix3D(Math.cos(teta), -Math.sin(teta), 0,

Math.sin(teta), Math.cos(teta), 0,

0, 0, 1);

var vecOut = vec.Rotate(mat); // vecOut = [0, 1, 0]

**Пример:**

vec = new Vector3D(1, 0, 0);

var teta = Math.PI/2;

var mat = new Matrix3D( Math.cos(teta), Math.sin(teta), 0,

0, 1, 0,

-Math.sin(teta), 0, Math.cos(teta));

var vecOut = vec.Rotate(mat); // vecOut = [0, 0, -1]

**Пример:**

vec = new Vector3D(0, 1, 0);

var teta = Math.PI/2;

var mat = new Matrix3D(1, 0, 0,

0, Math.cos(teta), -Math.sin(teta),

0, Math.sin(teta), Math.cos(teta));

var vecOut = vec.Rotate(mat); // vecOut = [0, 0, 1]

**function Matrix3D(m0x, m0y, m0z, m1x , m1y, m1z, m2x, m2y, m2z)**

Функция-конструктор трехмерных матриц Matrix3D.

m0x m1x m2x

m0y m1y m2y

m0z m1z m2z

**Пример:**

var mat = new Matrix3D(0,1,2, 3,4,5, 6,7,8);

// mat = [0,1,2]

// [3,4,5]

// [6,7,8]

**Пример:**

var mat = new Matrix3D();

// mat = [undefined, undefined, undefined]

// [undefined, undefined, undefined]

// [undefined, undefined, undefined]

**Matrix3D.prototype.Det = function()**

Функция возвращает величину определителя матрицы.

**Пример:**

var mat = new Matrix3D(-1, 2, -3,

4, 5, 6,

7, 8, -9);

var det = mat.Det(); // det = 258

**Пример:**

var mat = new Matrix3D();

var det = mat.Det(); // det = null

**Matrix3D.prototype.Inverse = function()**

Инверсия исходной матрицы this. Функция возвращает обратную матрицу к матрице this.

**Пример:**

var mat1 = new Matrix3D(1, 2, 3,

4, 5, 6,

7, -8, -9);

// обратная матрица

var inv1 = mat1.Inverse();

// умножение обратной матрицы на исходную дает единичную матрицу

var mat = inv1.MultiplyTwoMatrices(mat1);

// mat = [1, 0, 0]

// [0, 1, 0]

// [0, 0, 1]

**Пример:**

// детерминат матрицы mat равен 0

var mat = new Matrix3D(1,1,1, 1,1,1, 1,1,1); // det = 0

var inv = mat.Inverse(); // inv = null

**Matrix3D.prototype.MultiplyMatrixVector = function(vector)**

Возвращает вектор, полученный путем умножения матрицы this на вектор vector.

vecOut = this \* vector

**Пример:**

vec = new Vector3D(1, 0, 0);

var teta = Math.PI/2;

var mat = new Matrix3D(Math.cos(teta), -Math.sin(teta), 0,

Math.sin(teta), Math.cos(teta), 0,

0, 0, 1);

var vecOut = mat.MultiplyMatrixVector(vec); // vecOut = [0, 1, 0]

**Matrix3D.prototype.MultiplyTwoMatrices = function(mat)**

Возвращает матрицу, полученную путем умножения матрицы this на матрицу mat.

matOut = this \* mat

**Пример:**

var mat1 = new Matrix3D(1, 2, 3,

5, 6, 7,

0, 1, 2);

var mat2 = new Matrix3D(0.5, 6, 7,

4, 1, 2,

3, 2, 1);

var out = mat1.MultiplyTwoMatrices(mat2);

// out = [17.5, 14, 14]

// [47.5, 50, 54]

// [ 10, 5, 4]

**Matrix3D.prototype.Transpose = function(mat)**

Возвращает транспонированную матрицу mat.

**Пример:**

var mat = new Matrix3D(1, 2, 3,

4, 5, 6,

7, 8, 9);

var tr = mat.Transpose(mat); // tr = [1, 4, 7]

// [2, 5, 8]

// [3, 6, 9]

**Matrix3D.prototype.MatrixSetVectorInColumn = function(vec0, vec1, vec2)**

Устанавливает значения матрицы по столбцам, где vec0, vec1, vec2 - 3D векторы

**Пример:**

var vec0 = new Vector3D(1, 2, 3);

var vec1 = new Vector3D(4, 5, 6);

var vec2 = new Vector3D(7, 8, 9);

var mat = new Matrix3D();

var out = mat.MatrixSetVectorInColumn(vec0, vec1, vec2);

// out = [1, 4, 7]

// [2, 5, 8]

// [3, 6, 9]

**Matrix3D.prototype.MatrixSetVectorInRow = function(vec0, vec1, vec2)**

Устанавливает значения матрицы по рядам, где vec0, vec1, vec2 - 3D векторы

**Пример:**

var vec0 = new Vector3D(1, 2, 3);

var vec1 = new Vector3D(4, 5, 6);

var vec2 = new Vector3D(7, 8, 9);

var mat = new Matrix3D();

var out = mat.MatrixSetVectorInRow(vec0, vec1, vec2);

// out = [1, 2, 3]

// [4, 5, 6]

// [7, 8, 9]

**Matrix3D.prototype.UnitaryMatrix = function()**

Функция возвращает единичную матрицу и делает единичной матрицу this.

**Пример:**

var mat = new Matrix3D();

var unitMat = mat.UnitaryMatrix();

// mat = [1, 0, 0]

// [0, 1, 0]

// [0, 0, 1]

// unitMat = [1, 0, 0]

// [0, 1, 0]

// [0, 0, 1]

**Matrix3D.prototype.InvertMatrixMultiply = function (vector)**

Функция возвращает инвертированную матрицу this умноженную на вектор vector. Данная функция используется в функции IntersectionThreePlanes.

**Пример:**

var mat = new Matrix3D(-3, 0, 2,

3, -1, 0,

-2, 0, 1);

var vec = new Vector3D(1, 2, 3);

var vecOut = mat.InvertMatrixMultiply(vec);

// vecOut = [-5, -17, -7]

// инвертированная матрица = [1, 0, -2]

// [3, -1, -6]

// [2, 0, -3]

**Matrix3D.prototype.RotX = function(angle)**

Матрица поворота вокруг оси X на угол angle

**Пример:**

var mat = new Matrix3D(); // // создание (объявление) матрицы

mat.RotX(Math.PI/2); // заполнение компонент матрицы

// поворот вектора [0, 1, 0] вокруг оси X на 90°

var vecOut = mat.MultiplyMatrixVector(new Vector3D(0, 1, 0));

// vecOut = [0, 0, 1]

**Matrix3D.prototype.RotY = function(angle)**

Матрица поворота вокруг оси Y на угол angle

**Пример:**

var mat = new Matrix3D(); // создание (объявление) матрицы

mat.RotY(Math.PI/2); // заполнение компонент матрицы

// поворот вектора [1, 0, 0] вокруг оси Y на 90°

var vecOut = mat.MultiplyMatrixVector(new Vector3D(1, 0, 0));

// vecOut = [0, 0, -1]

**Matrix3D.prototype.RotZ = function(angle)**

Матрица поворота вокруг оси Z на угол angle

**Пример:**

var mat = new Matrix3D(); // создание (объявление) матрицы

mat.RotZ(Math.PI/2); // заполнение компонент матрицы

// поворот вектора [1, 0, 0] вокруг оси Z на 90°

var vecOut = mat.MultiplyMatrixVector(new Vector3D(1, 0, 0));

// vecOut = [0, 1, 0]

**Matrix3D.prototype.SetRotateMatrix = function(vector1, vector2, vector3**)

Установка компонент матрицы вращения (установка компонент матрицы this) по трем векторам vector1, vector2 и vector3.

**Пример:**

var vec1 = new Vector3D(1, 1, 0);

var vec2 = new Vector3D(0, 2, 2);

var vec3 = new Vector3D(3, 0, 3);

var mat = new Matrix3D();

mat.SetRotateMatrix(vec1, vec2, vec3);

// mat = [0.70710.., 0.70710.., 0]

// [0, 0.70710.., 0.70710..]

// [0.70710.., 0, 0.70710..]

**function Point3D(x, y, z)**

Функция-конструктор 3D точки Point3D, где x, y, z - координаты точки.

**Пример:**

var point1 = new Point2D(2, 1, 3);

var point2 = new Point3D();

// point2 = [undefined, undefined, undefined]

**Point3D.prototype.Add = function(point)**

Функция возвращает точку, являющюся суммой двух точек this и point.

**Пример:**

var pt1 = new Point3D(1, 2, 3);

var pt2 = new Point3D(4, -3, -7);

var ptOut = pt1.Add(pt2); // ptOut = [5, -1, -4]

**Point3D.prototype.Translate = function(vector)**

Возвращает точку this перенесенную на вектор vector

**Пример:**

var pt = new Point3D(1, 2, 3);

var vec = new Vector3D(4, -3, -7);

var ptOut = pt.Translate(vec); // ptOut = [5, -1, -4]

**Point3D.prototype.Distance = function(point)**

Возвращает расстояние между точкой this и точкой point.

**Пример:**

var pt1 = new Point3D(1, 1, 1);

var pt2 = new Point3D(2, 2, 2);

var dist = pt1.Distance(pt2);

// dist = 1.7320508075688772

**Point3D.prototype.Rotate = function(rot)**

Функция возвращает точку this, повернутую матрицей rot.

**Пример:**

var mat = new Matrix3D(); // создание (объявление) матрицы

mat.RotY(Math.PI/2); // заполнение компонент матрицы поворота

// поворот точки с координатами [1, 0, 0] вокруг оси Y на 90°

var pt = new Point3D(1, 0, 0);

var ptOut = pt.Rotate(mat);

// ptOut = [0, 0, -1]

**function SquareTriangle(point1, point2, point3)**

Функция возвращает площадь треугольника заданного точками point1, point2, и point3

**Пример:**

var pt1 = new Point3D(1,0,0);

var pt2 = new Point3D(0,1,0);

var pt3 = new Point3D(0,0,1);

var square = SquareTriangle(pt1, pt2, pt3);

// square = 0.8660254037844386

**function Line3D(point1, point2)**

Функция создает новую прямую в пространстве по двум точкам point1 и point2. Создаваемая прямая определяется своими направляющими косинусоми и точкой,принадлежащей прямой.

**Пример:**

var line1 = new Line3D();

// directCos = [undefined, undefined, undefined]

// pointOnLine = [undefined, undefined, undefined]

var line2 = new Line3D(new Point3D(1, 1, 1), new Point3D(1, 0, 0));

// directCos = [0, -0.7071067811865475, -0.7071067811865475]

// pointOnLine = [1, 1, 1]

**Line3D.prototype.CreateLineVectorPoint = function(vector, point)**

Функция создает прямую в пространстве направленную вдоль вектора vector и проходящую через точку point.

**Пример:**

var vec = new Vector3D(1, 1, 1);

var point = new Point3D(1, 0, 0);

var line = new Line3D();

var lineOut = line.CreateLineVectorPoint(vec, point);

// lineOut = line

// directCos = [0.577.., 0.577.., 0.577..]

// pointOnLine = [1, 0, 0]

**Line3D.prototype.Rotate = function(rot)**

Функция вращает прямую this в пространстве посредством матрицы rot

**Пример:**

var mat = new Matrix3D(); // создание (объявление) матрицы

mat.RotY(Math.PI/2); // заполнение компонент матрицы поворота для

// поворота прямой line вокруг оси Y на 90°

var line = new Line3D(new Point3D(1, 0, 0), new Point3D(0, 0, 0));

// directCos = [-1, 0, 0]

// pointOnLine = [1, 0, 0]

line.Rotate(mat);

// directCos = [0, 0, 1]

// pointOnLine = [0, 0, -1]

**Line3D.prototype.Angle = function(line)**

Функция возвращает угол между векторами определяющими направления прямой this   
и прямой line.

**Пример:**

var line1 = new Line2D(new Point3D(0, 0, 0), new Point2D(0, 1, 1));

var line2 = new Line2D(new Point3D(0, 0, 0), new Point2D(1, 1, 1));

var degreeAng12 = (180 / Math.PI) \* line1.Angle(line2);

// degreeAng12 = -45°

**function Plane3D(norm, dist)**

Функция-конструктор трехмерной плоскости.

Плоскость задается вектором нормали к плоскости norm и расстоянием dist от начала координат. Данная функция используется только для создания заготовки (объявления) плоскости. Конкретные параметры плоскости определяются в дальнейшем такими функциями как CreatePlaneThreePoints, CreatePlaneNormalVectorPoint, CreatePlaneVectorTwoPoints, CreateInclinePlane.

**Пример:**

var plane = new Plane3D();

// directCos = [undefined, undefined, undefined]

// distOXYZ = undefined

**Plane3D.prototype.Normer = function()**

Нормировка уравнения плоскости. Данная функция используется внутри функций CreatePlaneThreePoints, CreatePlaneNormalVectorPoint, CreatePlaneVectorTwoPoints, CreateInclinePlane.

**Plane3D.prototype.CreatePlaneThreePoints = function(point1, point2, point3)**

Функция создает плоскость проходящую через три точки point1, point2 и point3.

**Пример:**

var pt1 = new Point3D(1, 5, 2);

var pt2 = new Point3D(4, 2, 6);

var pt3 = new Point3D(3, 1, 3);

var plane = new Plane3D();

plane.CreatePlaneThreePoints(pt1, pt2, pt3);

// directCos = [0.8571.., 0.3296.., -0.3956..]

// distOXYZ = 1.7143892..

**Plane3D.prototype.CreatePlaneNormalVectorPoint = function(vector, point)**

**Пример:**

Создание плоскости по ее нормальному вектору и точке, принадлежащей плоскости.

var vec = new Vector3D(1, 1, 1);

var pt = new Point3D(1, 0, 0);

var plane = new Plane3D();

plane.CreatePlaneNormalVectorPoint(vec, pt);

// directCos = [0.5773.., 0.5773.., 0.5773..]

// distOXYZ = 0.5773502691896258

**Plane3D.prototype.CreatePlaneVectorTwoPoints = function(vector, point1, point2)**

Создание плоскости по вектору vec1, параллельному плоскости, и двум точкам point1 и point2, принадлежащим плоскости.

**Пример:**

var vec = new Vector3D(0, -1, 1);

var pt1 = new Point3D(0, 1, 0);

var pt2 = new Point3D(1, 0, 0);

var plane = new Plane3D();

var pl = plane.CreatePlaneVectorTwoPoints(vec, pt1, pt2);

// pl = plane

// directCos = [0.5773.., 0.5773.., 0.5773..]

// distOXYZ = 0.5773502691896258

**Plane3D.prototype.CreatePlaneNormalDistOXYZ = function(norm, dist)**

**Пример:**

Функция создает плоскость по направляющему вектору norm и расстоянию от начала координат dist.

var Z0 = new Vector3D(0, 0, 1);

var plane = new Plane3D();

plane.CreatePlaneNormalDistOXYZ(Z0, 4.0);

// directCos = [0, 0, 1]

// distOXYZ = 4

**Plane3D.prototype.IntersectionThreePlanes = function(plane2, plane3)**

Возвращает точку пересечения плоскости this с плоскостями plane2 и plane3.

**Пример:**

var plane1 = new Plane3D();

plane1.CreatePlaneThreePoints(new Point3D(1,1,0), new Point3D(1,1,1),   
 new Point3D(0,1,1));

var plane2 = new Plane3D();

plane2.CreatePlaneThreePoints(new Point3D(1,1,0), new Point3D(1,1,1),   
 new Point3D(1,0,1));

var plane3 = new Plane3D();

plane3.CreatePlaneThreePoints(new Point3D(1,1,1), new Point3D(1,0,1),   
 new Point3D(0,0,1));

var point = plane1.IntersectionThreePlanes(plane2, plane3);

// point = [1, 1, 1]

**Plane3D.prototype.VectorIntersectionTwoPlanes = function(plane)**

Функция возвращает вектор, имеющий направление прямой по которой пересекаются плоскости this и plane.

**Пример:**

var plane1 = new Plane3D();

plane1.CreatePlaneThreePoints(new Point3D(1,1,0), new Point3D(1,1,1),   
 new Point3D(0,1,1));

var plane2 = new Plane3D();

plane2.CreatePlaneThreePoints(new Point3D(1,1,1), new Point3D(1,0,1),   
 new Point3D(0,0,1));

var vec = plane1.VectorIntersectionTwoPlanes(plane2);

// vec = [1, 0, 0]

**Plane3D.prototype.Rotate = function(rot)**

Функция при помощи матрицы rot вращает плоскость this.

**Пример:**

var plane = new Plane3D();

plane.CreatePlaneThreePoints(new Point3D(1,1,1), new Point3D(1,0,1),   
 new Point3D(0,0,1));

// directCos = [0, 0, -1]

// distOXYZ = -1

var mat = new Matrix3D(); // создание (объявление) матрицы

mat.RotX(Math.PI/2); // заполнение компонент матрицы

// поворот плоскости plane вокруг оси X на 90°

plane.Rotate(mat);

// directCos = [0, 1, 0]

// distOXYZ = -1

var dist = plane.DistancePoint(new Point3D(0, 0, 0));

// dist = 1

**Plane3D.prototype.DistancePoint = function(point)**

Функция возвращает расстояние от точки point до плоскости this.

**Пример:**

var plane = new Plane3D();

// Создаем плоскость проходящую перпендикулярно оси OZ на расстоянии // равным 1 от начала координат.  
plane.CreatePlaneThreePoints(new Point3D(1,1,1), new Point3D(1,0,1),   
 new Point3D(0,0,1));

// Создаем две точки point1 и point2 находящиеся на одинаковом   
// расстоянии по абсолютной величине от плоскости plane но по разные   
// стороны от этой плоскости.

var point1 = new Point3D(0, 0, 3);

var dist1 = plane.DistancePoint(point1); // dist1 = -2

var point2 = new Point3D(0, 0, -1);

var dist2 = plane.DistancePoint(point2); // dist2 = 2

// Видно, что dist1 и dist2 различаются своими знаками.

**Plane3D.prototype.Slope = function()**

Функция возвращает угол между плоскостью this и плоскостью OXY.

**Пример:**

var plane1 = new Plane3D();

plane1.CreatePlaneThreePoints(new Point3D(1,1,1),   
 new Point3D(1,0,1),   
 new Point3D(0,0,1));

var ang1 = (180 / Math.PI) \* plane1.Slope();

// ang1 = 0°

var plane2 = new Plane3D();

plane2.CreatePlaneThreePoints(new Point3D(0, 0, 0),   
 new Point3D(0.5, 0, 1),   
 new Point3D(1, 1, 1));

var ang2 = (180 / Math.PI) \* plane2.Slope();

// ang2 = 65.90515744788931°

var plane3 = new Plane3D();

plane3.CreatePlaneThreePoints(new Point3D(0, 0, 0),   
 new Point3D(-0.5, 0, -1),   
 new Point3D(1, 1, -1));

var ang3 = (180 / Math.PI) \* plane3.Slope();

// ang3 = -74.49864043306299°

**Plane3D.prototype.Angle = function(plane)**

Функция возвращает угол между плоскостью this и плоскостью plane.

**Пример:**

var plane1 = new Plane3D();

plane1.CreatePlaneThreePoints(new Point3D(0, 0, 0),   
 new Point3D(1,0,1), new Point3D(1,1,1));

var plane2 = new Plane3D();

plane2.CreatePlaneThreePoints(new Point3D(0, 0, 0),   
 new Point3D(1,0,1), new Point3D(1,0,0));

var ang = (180 / Math.PI) \* plane1.Angle(plane2);

// ang = 90°

**Plane3D.prototype.Normal = function()**

Функция возвращает вектор перпендикулярный к плоскости this.

**Пример:**

var plane = new Plane3D();

plane.CreatePlaneThreePoints(new Point3D(1,1,3), new Point3D(1,0,3),   
 new Point3D(0,0,3));

var normal = plane.Normal(); // normal = [0, 0, -1]

**Plane3D.prototype.Distance = function()**

Функция возвращает расстояние от плоскости this до начала координат.

**Пример:**

var plane = new Plane3D();

plane.CreatePlaneThreePoints(new Point3D(1,1,3), new Point3D(1,0,3),   
 new Point3D(0,0,3));

var dist = plane.Distance(); // dist = -3

**Line3D.prototype.IntersectionLinePlane = function(plane)**

Функция возвращает точку пересечения прямой this с плоскостью plane.

**Пример:**

var plane = new Plane3D();

plane.CreatePlaneThreePoints(new Point3D(1,1,3), new Point3D(1,0,3),   
 new Point3D(0,0,3));

var line = new Line3D(new Point3D(2, 3, -4), new Point3D(2, 3, -6));

var point = line.IntersectionLinePlane(plane);

// point = [2, 3, 3]

**Plane3D.prototype.CreateInclinePlane = function(angle, point1, point2, point3)**

Данная функция создает плоскость наклоненную относительно плоскости OXY на угол angle. Точки point1 и point2 должны иметь одинаковое значение координаты z (иными словами pt1[2] = pt2[2]). Рассчитывается двумерный вектор norm2d и затем определяется плоскость имеющая наклон angle к плоскости OXY.   
 Данная функция делает то же самое что и функция Facet, но в ней в отличие от функции Facet, используется наследование через прототипы.   
 Для лучшего понимания работы функции ниже приведен ее текст:

Plane3D.prototype.CreateInclinePlane = function(angle, pt1, pt2, pt3) {   
 var norm2d = new Vector2D(pt1[1] - pt2[1], pt2[0] - pt1[0]);  
 norm2d.Normer();  
 var x = Math.sin(angle) \* norm2d[0];  
 var y = Math.sin(angle) \* norm2d[1];  
 var z = Math.cos(angle);  
 var normPlaneVector = new Vector3D(x, y, z);  
 normPlaneVector.Normer();  
 this.CreatePlaneNormalVectorPoint(normPlaneVector, pt3);  
 this.Normer();  
}

Рассчитывается двумерный вектор norm2d и затем определяется плоскость имеющая наклон angle к плоскости OXY. Рассчитывается двумерный вектор norm2d, который направлен перпендикулярно к отрезку, соединяющему точки pt1 и pt2. После этого вычисляется 3D вектор normPlaneVector, который имеет угол наклона к горизонтальной плоскости (OXY) равный величине угла angle. Проекции этого вектора на горизонтальную плоскость определяется вектором norm2d и синусом угла angle. Для возвращаемой функцией Facet плоскости вектор normPlaneVector принимается в качестве нормального вектора. Этой плоскости принадлежит точка pt3. Следует учесть, что вектор normPlaneVector может быть повернут на угол angle относительно отрезка, соединяющего точки pt1 и pt2 двумя способами. В качестве примера можно взять построение граней короны огранки **Octagon** примыкающих к рундисту. В этой огранке эти грани направлены как бы "внутрь рундиста". Поэтому задание точек pt1 и pt2 в параметрах функции Facet выбирается вполне определенным способом - 15 и 8, 8 и 9, 9 и 10:

var planeA = new Plane3D();  
 planeA.CreateInclinePlane(-angleA, girdle[15], girdle[8], girdle[8]);  
 var planeB = new Plane3D();  
 planeB.CreateInclinePlane(-angleB, girdle[8], girdle[9], girdle[9]);  
 var planeC = new Plane3D();   
 planeC.CreateInclinePlane(-angleC, girdle[9], girdle[10], girdle[10]);

Если изменить выбор пар точек, например, осуществить выбор точек для грани planeB так, как показано ниже, то эта грань будет построена не так как ожидалось.

var planeB = new Plane3D();  
 planeB.CreateInclinePlane(-angleB, girdle[9], girdle[8], girdle[9]);

Следует запустить программу **Octagon\_text.html** в режиме Crown-Gd.-Pav. для того, чтобы увидеть нумерацию вершин огранки и посмотреть соответствующую часть текста программы **octagon\_verts.js** относящуюся к построению соответствующих граней короны.

**function Facet(angle, point1, point2, point3)**

Предполагается, что точки pt1 и pt2 имеют одинаковую величину координаты z (иными словами pt1[2] = pt2[2]). Рассчитывается двумерный вектор norm2d и затем определяется возвращаемая плоскость имеющая наклон angle к плоскости OXY.   
 Данная функция делает то же самое что и функция CreateInclinePlane, но в ней не используется наследование через прототипы.

**Пример:**

var pt1 = new Point3D(0,0,4);

var pt2 = new Point3D(2,1,4)

var pt3 = new Point3D(0.5, 0.5, 6.0)

var plane = Facet(Math.PI/4, pt1, pt2, pt3);

// directCos = [ -0.31622.., 0.63245.., -0.70710..]

// distOXYZ = -4.084526804110867