Math\Physics Core Documentation

**Class Vector2D**

Methods:

Vector2D()

return type: none

parameters : none

Vector2D()

return type: none

parameters: Vector2D&

Copy constructor

Vector2D()

return type: none

parameters: Scalar

Initialize with an array

Vector2D()

return type: none

parameters: double, double

Operator[]

return type: Scalar&

parameters: int

Use to both read and write elements, just like a normal array

Operator[]

return type: Scalar

parameters: int

Use to read elements from const vectors

Length()

return type: Scalar

parameters: none

Normalize()

return type: none

parameters: none

Modifies the vector to be unit length

Normalized()

return type: Vector2D

parameters: none

Returns a new vector that is unit length

Operator-()

return type: Vector2D

parameters: none

Return negative of a vector

Operator\*=()

return type: none

parameters: Scalar

Operator/=()

return type: none

parameters: Scalar

Operator\*()

return type: Vector2D

parameters: Scalar

For the case when the operand order is Vector \* Scalar

Operator/()

return type: Vector2D

parameters: Scalar

For the case when the operand order is Vector / Scalar

Operator\*()

return type: Vector2D

parameters: Scalar, Vector2D&

For the case when the operand order is Scalar \* Vector

Operator+()

return type: Vector2D

parameters: Vector2D&

Vector addition

Operator-()

return type: Vector2D

parameters: Vector2D&

Vector subtraction

Operator +=()

return type: none

parameters: Vector2D&

Vector addition

Operator -=()

return type: none

parameters: Vector2D&

Vector subtraction

Operator\*()

return type: Scalar

parameters: Vector2D&

Dot-product

Orthogonal()

return type: Vector2D

parameters: none

Construct a vector that is orthogonal (perpendicular) to the given vector

Operator==()

return type: bool

parameters: Vector2D&

Operator !=()

return type: bool

parameters: Vector2D&

Zero\_Vector()

return type: Vector2D&

parameters: none

Use this to test whether or not a vector == zero vector

**Class Vector3D**

Methods:

Vector3D()

return type: none

parameters : none

Vector3D()

return type: none

parameters: Vector3D&

Copy constructor

Vector3D()

return type: none

parameters: Scalar

Initialize with an array

Vector3D()

return type: none

parameters: double, double, double

Operator[]

return type: Scalar&

parameters: int

Use to both read and write elements, just like a normal array

Operator[]

return type: Scalar

parameters: int

Use to read elements from const vectors

Length()

return type: Scalar

parameters: none

Normalize()

return type: none

parameters: none

Modifies the vector to be unit length

Normalized()

return type: Vector3D

parameters: none

Returns a new vector that is unit length

Operator-()

return type: Vector3D

parameters: none

Negate a vector

Operator\*()

return type: Vector3D

parameters: Scalar

For the case when the operand order is Vector \* Scalar

Operator\*()

return type: Vector3D

parameters: Scalar, Vector3D&

For the case when the operand order is Scalar \* Vector

Operator/()

return type: Vector3D

parameters: Scalar

For the case when the operand order is Vector / Scalar

Operator\*=()

return type: none

parameters: Scalar

Operator/=()

return type: none

parameters: Scalar

Operator+()

return type: Vector3D

parameters: Vector3D&

Vector addition

Operator-()

return type: Vector3D

parameters: Vector3D&

Vector subtraction

Operator +=()

return type: none

parameters: Vector3D&

Vector addition

Operator -=()

return type: none

parameters: Vector3D&

Vector subtraction

Operator\*()

return type: Scalar

parameters: Vector3D&

Dot-product

Cross\_Product()

return type: Vector3D

parameters: Vector3D&

Construct a vector that is orthogonal (perpendicular) to the given vector

Operator%()

return type: Vector3D

parameters: Vector3D&

Overload for cross product

Operator==()

return type: bool

parameters: Vector3D&

Operator !=()

return type: bool

parameters: Vector3D&

Zero\_Vector()

return type: Vector3D&

parameters: none

Use this to test whether or not a vector == zero vector

**Class Vector4D**

Methods:

Vector4D()

return type: none

parameters : none

Vector4D()

return type: none

parameters: Vector4D&

Copy constructor

Vector4D()

return type: none

parameters: Scalar

Initialize with an array

Vector4D()

return type: none

parameters: double, double, double, double

Vector4D()

return type: none

parameters: double, double, double

Vector4D()

return type: none

parameters: Vector3D, Scalar

ToVector3D()

return type: Vector3D

parameters: none

Returns the 1st 3 components as a Vector3D

Operator[]

return type: Scalar&

parameters: int

Use to both read and write elements, just like a normal array

Operator[]

return type: Scalar

parameters: int

Use to read elements from const vectors

Length()

return type: Scalar

parameters: none

Length3D()

return type: Scalar

parameters: none

Length of the normal

Normalize()

return type: none

parameters: none

Modifies the vector to be unit length

Normalize3D()

return type: none

parameters: none

Normalize the normal only

Normalized()

return type: Vector4D

parameters: none

Returns a new unit vector

Operator\*=()

return type: none

parameters: Scalar

Operator\*()

return type: Vector4D

parameters: Scalar

For the case when the operand order is Vector \* Scalar

Operator\*()

return type: Vector4D

parameters: Scalar, Vector4D&

For the case when the operand order is Scalar \* Vector

Operator/=()

return type: none

parameters: Scalar

Operator/()

return type: Vector4D

parameters: Scalar

For the case when the operand order is Vector / Scalar

Operator +=()

return type: none

parameters: Vector4D&

Vector addition

Operator -=()

return type: none

parameters: Vector4D&

Vector subtraction

Operator+()

return type: Vector4D

parameters: Vector4D&

Vector addition

Operator-()

return type: Vector4D

parameters: Vector4D&

Vector subtraction

Operator\*()

return type: Scalar

parameters: Vector4D&

Dot-product

Operator-()

return type: Vector4D

parameters: none

Negate a vector

Cross\_Product()

return type: Vector4D

parameters: Vector4D&

Construct a vector that is orthogonal (perpendicular) to the given vector

Operator%()

return type: Vector4D

parameters: Vector4D&

Overload for cross product

Operator%()

return type: Vector3D

parameters: Vector3D&

Cross product with a Vector3D

Cross\_Product()

return type: Vector4D

parameters: Vector4D, Vector4D

Operator==()

return type: bool

parameters: Vector4D&

Operator !=()

return type: bool

parameters: Vector4D&

Zero\_Vector()

return type: Vector4D&

parameters: none

Use this to test whether or not a vector == zero vector

**Class Matrix2D**

Methods:

Matrix2D()

return type: none

parameters : none

Set to the zero matrix

Matrix2D()

return type: none

parameters : Vector2D, Vector2D

Set matrix columns to 2 basis vectors

Identity()

return type: none

parameters : none

Set to the identity matrix

Scaling()

return type: none

parameters : Vector2D

Set to a scaling matrix

Rotation()

return type: none

parameters: double

Set to a rotation matrix

Shearing()

return type: none

parameters: Vector2D

Set to a shear matrix

Scale()

return type: none

parameters: Scalar

Multiply by a new scaling matrix

Scale()

return type: none

parameters: Vector2D

Multiply by a new scaling matrix

Rotate()

return type: none

parameters: double

Multiply by a new rotation matrix

Shear()

return type: none

parameters: Vector2D

Multiply by a new shearing matrix

Operator+()

return type: Matrix2D

parameters: Matrix2D&

Matrix addition

Operator-()

return type: Matrix2D

parameters: Matrix2D&

Matrix subtraction

Operator\*()

return type: Matrix2D

parameters: Matrix2D&

Matrix multiplication

Operator \*=()

return type: none

parameters: Matrix2D&

Multiply a matrix by another matrix

Operator\*()

return type: Vector2D

parameters: Vector2D&

Matrix \* Vector. Used to transform vertices

Operator\*=()

return type: none

parameters: Vector2D&, Matrix2D&

Transposed-vector \* matrix. Used to transform normals

Determinant()

return type: Scalar

parameters: none

Inverse()

return type: Matrix2D

parameters: none

Returns a new matrix that is the inverse of this matrix

Invert()

return type: none

parameters: none

Inverts this matrix in place

Operator==()

return type: bool

parameters: Matrix2D&

Used for comparison

Operator!=()

return type: bool

parameters: Matrix2D&

Used for comparison

Identity\_Matrix()

return type: Matrix2D&

parameters: none

Return an identity matrix to be used in comparisons with other matrices

**Class Matrix3D**

Methods:

Matrix3D()

return type: none

parameters : none

Set to the zero matrix

Matrix3D()

return type: none

parameters : Vector3D, Vector3D, Vector3D

Set matrix columns to 3 basis vectors

Basis()

return type: Vector3D

parameters: int

Get a basis vector (column)

Identity()

return type: none

parameters : none

Set to the identity matrix

Scaling()

return type: none

parameters: Scalar

Set to a new scaling matrix

Scaling()

return type: none

parameters : Vector3D

Set to a scaling matrix

Rotation()

return type: none

parameters: double, int

Set to a rotation matrix

Shearing()

return type: none

parameters: Vector2D, int

Set to a shear matrix

Scale()

return type: none

parameters: Scalar

Multiply by a new scaling matrix

Scale()

return type: none

parameters: Vector3D

Multiply by a new scaling matrix

Rotate()

return type: none

parameters: double, int

Multiply by a new rotation matrix

Shear()

return type: none

parameters: Vector2D, int

Multiply by a new shearing matrix

Operator+()

return type: Matrix3D

parameters: Matrix3D&

Matrix addition

Operator-()

return type: Matrix3D

parameters: Matrix3D&

Matrix subtraction

Operator\*()

return type: Matrix3D

parameters: Matrix3D&

Matrix multiplication

Operator \*=()

return type: none

parameters: Matrix3D&

Multiply a matrix by another matrix

Operator\*()

return type: Vector3D

parameters: Vector3D&

Matrix \* Vector. Used to transform vertices

Operator\*=()

return type: none

parameters: Vector3D&, Matrix3D&

Transposed-vector \* matrix. Used to transform normals

Determinant()

return type: Scalar

parameters: none

Inverse()

return type: Matrix3D

parameters: none

Returns a new matrix that is the inverse of this matrix

Invert()

return type: none

parameters: none

Inverts this matrix in place

Transpose()

return type: Matrix3D

parameters: none

Trace()

return type: Scalar

parameters: none

Sum of diagonal elements

Operator==()

return type: bool

parameters: Matrix3D&

Used for comparison

Operator!=()

return type: bool

parameters: Matrix3D&

Used for comparison

Identity\_Matrix()

return type: Matrix3D&

parameters: none

Return an identity matrix to be used in comparisons with other matrices

Determinant2D()

return type: Scalar

parameters: int, int

**Class Matrix4D**

Methods:

Matrix4D()

return type: none

parameters : none

Set to the zero matrix

Matrix4D()

return type: none

parameters : Vector3D, Vector3D, Vector3D

Set matrix columns to 3 basis vectors, 4th column will be [0 0 0 1]

Matrix4D()

return type: none

parameters: Vector4D, Vector4D, Vector4D, Vector4D

Set matrix columns to 4 basis vectors

Identity()

return type: none

parameters : none

Set to the identity matrix

Scaling()

return type: none

parameters: Scalar

Set to a new scaling matrix

Scaling()

return type: none

parameters : Vector3D

Set to a scaling matrix

Rotation()

return type: none

parameters: double, int

Set to a rotation matrix

Shearing()

return type: none

parameters: Vector2D, int

Set to a shear matrix

Translation()

return type: none

parameters: Vector3D&

Set to a 3D translation matrix

Scale()

return type: none

parameters: Scalar

Multiply by a new scaling matrix

Scale()

return type: none

parameters: Vector3D

Multiply by a new scaling matrix

Rotate()

return type: none

parameters: double, int

Multiply by a new rotation matrix

Shear()

return type: none

parameters: Vector2D, int

Multiply by a new shearing matrix

Translate()

return type: none

parameters: Vector3D&

Multiply by a new 3D translation matrix

Translate()

return type: none

parameters: Vector4D&

Multiply by a new 3D translation matrix

Operator+()

return type: Matrix4D

parameters: Matrix4D&

Matrix addition

Operator-()

return type: Matrix4D

parameters: Matrix4D&

Matrix subtraction

Operator+=()

return type: none

parameters: Matrix4D&

Matrix addition

Operator -=()

return type: none

parameters: Matrix4D&

Matrix subtraction

Operator\*()

return type: Matrix4D

parameters: Matrix4D&

Matrix multiplication

Operator \*=()

return type: none

parameters: Matrix4D&

Multiply a matrix by another matrix

Operator\*()

return type: Vector4D

parameters: Vector4D&

Matrix \* Vector. Used to transform vertices

Operator\*=()

return type: none

parameters: Vector4D&, Matrix4D&

Transposed-vector \* matrix. Used to transform normals

Determinant()

return type: Scalar

parameters: none

Inverse()

return type: Matrix4D

parameters: none

Returns a new matrix that is the inverse of this matrix

Invert()

return type: none

parameters: none

Inverts this matrix in place

Transpose()

return type: Matrix4D

parameters: none

Trace()

return type: Scalar

parameters: none

Sum of diagonal elements

Operator==()

return type: bool

parameters: Matrix4D&

Used for comparison

Operator!=()

return type: bool

parameters: Matrix4D&

Used for comparison

Identity\_Matrix()

return type: Matrix4D&

parameters: none

Return an identity matrix to be used in comparisons with other matrices

Determinant2D()

return type: Scalar

parameters: int, int

Determinant3D()

return type: Scalar

parameters: int, int

**Class Point**

Methods:

Rectangular()

return type: Point

parameters: float, float

Rectangular coordinates

Polar()

return type: Point

parameters: float, float

Polar coordinates

Point()

return type: none

parameters: float, float

Set coordinates