

LinaJS Tutorial

Table of Content

Overview.....	1
Points and vectors.....	2
Matrices.....	4
BiTran2.....	4
Polygons.....	5
Further information.....	5

Overview

LinaJS (*'linear algebra for JS'*) is an object oriented math library for points, vectors, and homogeneous transformations in 2D space.

It was developed as math backend for [Arcade.js](#).

Details:

<https://cdn.rawgit.com/mar10/arcade-js/master/doc/jsdocs/lina.js/jsdoc/symbols/LinaJS.html>

You will find a detailed documentation of the LinaJS API at

<https://cdn.rawgit.com/mar10/arcade-js/master/doc/jsdocs/lina.js/jsdoc/index.html>

It comes as part of ArcadeJS, an Open Source project [released under MIT](#), hosted at

<https://github.com/mar10/arcade-js/>

Constructors and arguments

Constructors (and some other methods) accept different argument types.

For example:

```
// Default constructor
var m1 = new Matrix3(); // Identity matrix
// Single values for x and y
var pt1 = new Point2(3, 4);
// Data objects
var pt2 = new Point2({x: 3, y: 4});
// Copy constructor
var pt3 = new Point2(pt1);
```

In-place operations, chaining, and transformed copies

Most transformation methods work 'in-place', thus modifying the object itself:

```
pt1.translate(1, 2);
```

Most methods also return the object instance, so transformations can be 'chained':

```
pt1.translate(1, 2).rotate(Math.PI).scale(1.5);
```

If we want a new translated instance instead of modifying the original one, we must create a copy

before:

```
var pt2 = pt1.copy().translate(1, 2);
```



Note pitfall:

Since modifications are in-place by default, the following example will **not** work as expected:

```
var pt1 = new Point2(0, 0);  
var pt2 = pt1.translate(1, 2); // <<-- WARNING pt1 is translated too
```

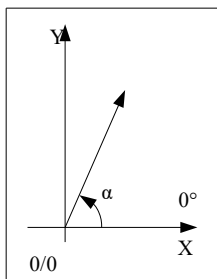
Use `.copy()` instead:

```
var pt2 = pt1.copy().translate(1, 2);
```

Here is another example that defines two matrices:

```
// Define transformation matrices  
var wc2cc = new Matrix3()  
  .translate(-vpa.x, -vpa.y)  
  .scale(ccWidth/vpa.width, -ccHeight/vpa.height)  
  .translate(0, ccHeight);  
var cc2wc = wc2cc.copy().invert();
```

Angles



Angles are specified in radians, starting at the positive x-axis (i.e. 0° is at three o'clock).

To rotate left by 90° we can do

```
pt1.rotate(0.5 * Math.PI);
```

or use the a conversion factor to convert from degree to radians

```
pt1.rotate(90 * LinaJS.D2R);
```

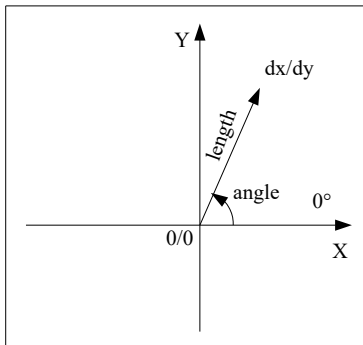
Points and vectors

Unlike some other libraries, LinaJS makes a difference between points and vectors:

	Point	Vector
Class	Point2	Vec2
Meaning	Represents a location.	Represents an offset (direction and distance).
Notation	(x, y)	(dx/dy)
Homogenous representation	$[x \ y \ 1]$	$[dx \ dy \ 0]$

Add - Operation	Adding a vector to a point yields a translated point. Two points cannot be added.	Adding two vectors yields another vector. A point cannot be added to a vector.
Rotation about arbitrary pivot	Moves the point to a different location and may change the distance to the center (0, 0).	Changes the direction, but not the length.

A 2-dimensional vector is implemented by the Vec2 class.



Here is an example, that adds two vectors, and creates a moved point from (1,1) by offset (2/3):

```
// Define transformation matrices
var ptOrg = new Point2(1, 1);
var ofs1 = new Vec2(2, 0);
var ofs2 = ofs1.copy().add(0, 3);
ptOrg.translate(ofs2);
```

Details on Point2 class:

<https://cdn.rawgit.com/mar10/arcade-js/master/doc/jsdocs/lina.js/jsdoc/symbols/Point2.html>

Details on Vec2 class:

<https://cdn.rawgit.com/mar10/arcade-js/master/doc/jsdocs/lina.js/jsdoc/symbols/Vec2.html>

Matrices

Matrices are used to store and accumulate transformations (R: rotation, T: translation, S: scale, ...).

They are especially efficient, when multiple transformations should be applied to a large number of objects. For example when transforming polygons from world coordinates to viewport coordinates.

```
// Define transformation matrices
var wc2cc = new Matrix3()
  .translate(-vpa.x, -vpa.y)
  .scale(ccwidth/vpa.width, -ccHeight/vpa.height)
  .translate(0, ccHeight);
// Transform points
for(var i=0; i<points.length; i++) {
  points[i].transform(wc2cc);
}
```

The Matrix3 class represent 2d homogenous transformation matrices:

$$\begin{bmatrix} R & R & 0 \\ R & R & 0 \\ Tx & Ty & 1 \end{bmatrix}$$

Internally stored as a float array with 9 elements:

$$[m_0 \ m_1 \ m_2 \ m_3 \ m_4 \ m_5 \ m_6 \ m_7 \ m_8]$$

interpreted as

$$\begin{bmatrix} m_0 & m_1 & m_2 \\ m_3 & m_4 & m_5 \\ m_6 & m_7 & m_8 \end{bmatrix}$$

Details:

<https://cdn.rawgit.com/mar10/arcade-js/master/doc/jsdocs/lina.js/jsdoc/symbols/Matrix3.html>

BiTran2

Maintain a 3x3 homogenous transformation that also maintains its own inversion, which may be more efficient then re-calculating the matrix inverse all the time:

<https://cdn.rawgit.com/mar10/arcade-js/master/doc/jsdocs/lina.js/jsdoc/symbols/BiTran2.html>

Polygons

The internal format is a flat array of float values.

Details:

<https://cdn.rawgit.com/mar10/arcade-js/master/doc/jsdocs/lina.js/jsdoc/symbols/Polygon2.html>

Further information

The LinaJS API is documented at

<https://cdn.rawgit.com/mar10/arcade-js/master/doc/jsdocs/lina.js/jsdoc/symbols/LinaJS.html>