# LO5 – Using Matrices to Represent Graphic Objects

2-dimensional objects – can be represented as a set of data points

These can be loaded into a matrix of size N×2 (for now) where N is the number of data points.

A graphic object can have 3 things done to it (transformations): For each point (x, y) of a graphic object that is being transformed, its new location is represented as (x', y').

1. Scaling – make the object bigger or smaller

To scale the object by some factor Sx and Sy,

x' = x \* Sx

y' = y \* Sy

1. Translation – move object from one point to another point

To translate the object by some amount Tx and Ty,

x' = x + Tx

y' = y + Ty

1. Rotation – rotate the object about some point  
   To rotate clockwise by an angle Θ (measured in radians)

x' = x \* cos Θ + y \* sin Θ

y' = -x \* sin Θ + y cos Θ

(Recall that to convert radians to degrees, radians = degrees \* π / 180)

We saw in our examples that scaling and rotation also moved the object. Rotation is done around the “world origin” (0, 0). We will need to pick a “local origin” to rotate an object about its own origin. To do so:

* Translate the object to the world origin
* Scale or rotate the object
* Translate back to the local origin

To perform successive operations on a set of data points, homogeneous coordinates are used.

In a 2-D set of data points, an additional column of 1’s is added

Thus the point (x, y) is represented as [x y 1]

The transformation matrices are:

We can multiply any set of points represented by homogeneous coordinates by the transformation matrix to get the transformed series of points.

Example: Let’s take the line from (1,1) to (2,2) and translate it by Tx = 2 and Ty = -1

Recall that to counter the side effects of scaling and rotation, we can translate the object to the world origin, scale or rotate, and then translate the object back to the local origin. We can do that by multiplying by 3 matrices: points \* T-x,-y \* S \* Tx,y or points \* T-x,-y \* R \* Tx,y

How can we reduce the work? Let’s examine rotation.

You should consider how to reduce the work for scaling. Recall that result = points \* T-x,-y \* S \* Tx,y