**Chapter 5**

**2D Transformations[[1]](#footnote-0)**

* Changing Position, shape, size, or orientation of an object on display is known as ***transformation***.
* **A transformation** is a function that takes a point (or vector) and maps it into another point (**or vector)**.
* Transformation is needed to observe objects in some other coordination **systems and animations.**
* A Cartographer can change the size of charts and topographical maps. So, if graphics images are coded as numbers, the numbers can be stored in memory. These numbers are modified by mathematical operations called Transformation.
* The purpose of using computers for drawing is to provide a facility to the user to view the object from different angles, enlarging or reducing the scale or shape of the object.

***Two essential aspects of transformation are given below:***

1. Each transformation is a single entity. It can be denoted by a unique name or symbol.
2. It is possible to combine two transformations, after connecting a single transformation is obtained, e.g., A is a transformation for ***translation***. The B transformation performs ***scaling***. The combination of two is C=AB. So C is obtained by concatenation property.

***There are two complementary points of view for describing object transformation.***

1. ***Geometric Transformation***: The object itself is transformed relative to the coordinate system or background. The mathematical statement of this viewpoint is defined by geometric transformations applied to each point of the object.
2. ***Coordinate Transformation***: The object is held stationary while the coordinate system is transformed relative to the object. This effect is attained through the application of coordinate transformations.

***An example that helps to distinguish these two viewpoints:***

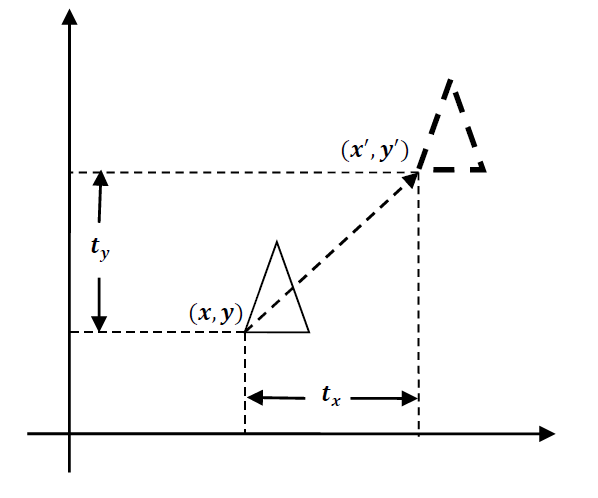
* The movement of an automobile against a scenic background we can simulate this by:
* Moving the automobile while keeping the background fixed- (Geometric Transformation)
* We can keep the car fixed while moving the background scenery- (Coordinate Transformation)
* **Types of Transformations:**

1. Translation
2. Scaling
3. Rotating
4. Reflection
5. Shearing

* **Basic Transformation**
* Basic transformation includes three transformations **Translation**, **Rotation**, and **Scaling**.
* These three transformations are known as basic transformations because with the combination of these three transformations we can obtain any transformation.
* **Four attributes of object that may be transformed:**
  + **Position** 🡪 Translation
  + **Size** 🡪 Scaling
  + **Orientation** 🡪 Rotation
  + **Shapes** 🡪 Shearing

# Translation

* It is the straight line movement of an object from one position to another. The object is positioned from one coordinate location to another.
* **Translation of point:**
* To translate a point from coordinate position (x, y) to another (x`, y`), we add algebraically the translation distances Tx and Ty to the original coordinate.

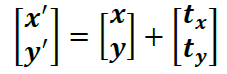


* We translate two dimensional point by adding translation distance 𝒕𝒙 and 𝒕𝒚 to the original coordinate position (𝒙, 𝒚) to move at new position (𝒙′, 𝒚′) as:

𝒙′ = 𝒙 + 𝒕𝒙 **&** 𝒚′ = 𝒚 + 𝒕𝒚

* Translation distance pair (𝒕𝒙, 𝒕𝒚) is called a ***Translation Vector*** or ***Shift*** ***Vector***.
* We can represent it into single matrix equation in column vector as;

𝑷′ = 𝑷 + 𝑻



* We can also represent it in row vector form as:

𝑷′ = 𝑷 + 𝑻

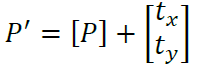
[𝒙′ 𝒚′] = [𝒙 𝒚] + [𝒕𝒙 𝒕𝒚]

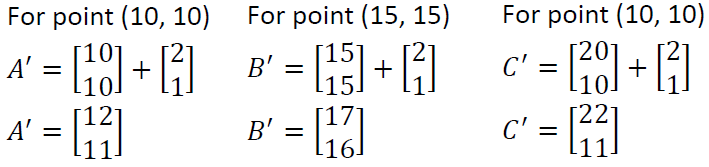
* Since column vector representation is standard mathematical notation and since many graphics packages like **GKS** and **PHIGS** use column vectors we will also follow column vector representation.
* Translation is a movement of objects without deformation. Every position or point is translated by the same amount. When the straight line is translated, then it will be drawn using endpoints.
* For translating a polygon, each vertex of the polygon is converted to a new position. Similarly, curved objects are translated. To change the position of the circle or ellipse its center coordinates are transformed, then the object is drawn using new coordinates.

***Example***: - Translate the triangle [A (10, 10), B (15, 15), C (20, 10)] 2 units in x direction and 1 unit in y direction.

***Solution***-We know that

𝑃′ = 𝑃 + 𝑇





Final coordinates after translation are [A’ (12, 11), B’ (17, 16), C’ (22, 11)].

# Rotation:

* It is a process of changing the angle of the object used to reposition the object along the circular path in the XY - plane.
* To generate a rotation, we specify a rotation angle 𝜽 and the position of the ***Rotation Point (Pivot Point)*** (𝒙𝒓, 𝒚𝒓) about which the object is to be rotated.
* **Types of Rotation:**

1. Clockwise
2. Counterclockwise

* The positive value of the pivot point (rotation angle) rotates an object in a ***counter-clockwise (anti-clockwise)*** direction.
* The negative value of the pivot point (rotation angle) rotates an object in a ***clockwise direction***.
* When the object is rotated, then every point of the object is rotated by the same angle.

**Straight Line:** Straight Line is rotated by the endpoints with the same angle and redrawing the line between new endpoints.

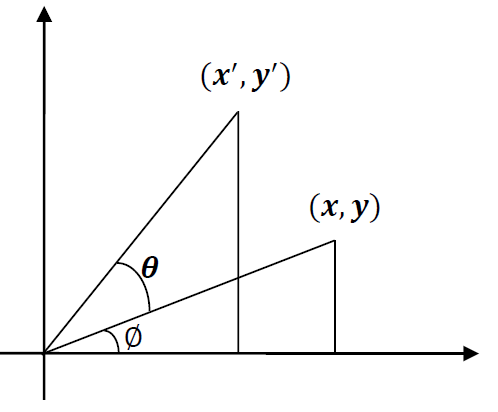
**Polygon:** Polygon is rotated by shifting every vertex using the same rotational angle.

**Curved Lines:** Curved Lines are rotated by repositioning of all points and drawing of the curve at new positions.

**Circle:** It can be obtained by center position by the specified angle.

**Ellipse:** Its rotation can be obtained by rotating the major and minor axis of an ellipse by the desired angle.

1. **We first find the equation of rotation when pivot point is at coordinate origin (𝟎, 𝟎).**



From the figure we can write.

𝒙 = 𝒓 𝐜𝐨𝐬 ∅

𝒚 = 𝒓 𝐬𝐢𝐧 ∅

and

𝒙′ = 𝒓 𝐜𝐨𝐬(𝜽 + ∅) = 𝒓 𝐜𝐨𝐬 ∅ 𝐜𝐨𝐬 𝜽 − 𝒓 𝐬𝐢𝐧 ∅ 𝐬𝐢𝐧 𝜽

𝒚′ = 𝒓 𝐬𝐢𝐧(∅ + 𝜽) = 𝒓 𝐜𝐨𝐬 ∅ 𝐬𝐢𝐧 𝜽 + 𝒓 𝐬𝐢𝐧 ∅ 𝐜𝐨𝐬 𝜽

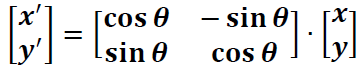
Now replace 𝒓 𝐜𝐨𝐬 ∅ with 𝒙 and 𝒓 𝐬𝐢𝐧 ∅ with 𝒚 in above equation.

𝒙′ = 𝒙 𝐜𝐨𝐬 𝜽 − 𝒚 𝐬𝐢𝐧 𝜽

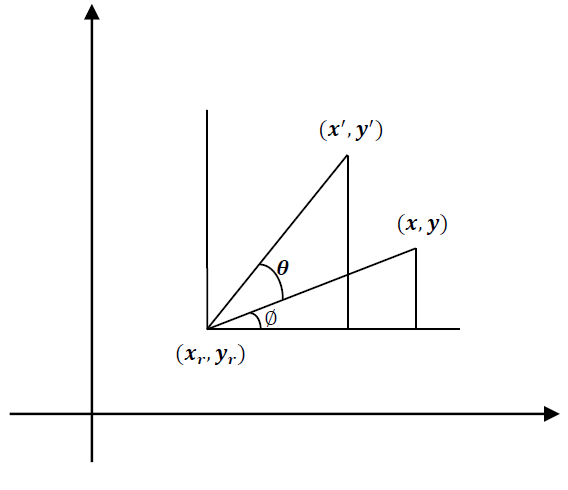
𝒚′ = 𝒙 𝐬𝐢𝐧 𝜽 + 𝒚 𝐜𝐨𝐬 𝜽

We can write it in the form of column vector matrix equation as;

𝑷′ = 𝑹 ∙ 𝑷



1. **Rotation about arbitrary points is illustrated in the figure below.**

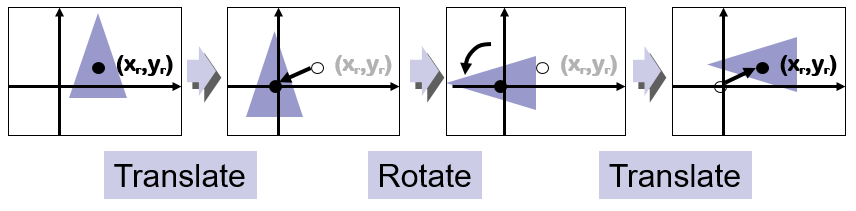


Transformation equation for rotation of a point about pivot point (𝒙𝒓, 𝒚𝒓) is:

𝒙′ = 𝒙𝒓 + (𝒙 − 𝒙𝒓) 𝐜𝐨𝐬 𝜽 − (𝒚 − 𝒚𝒓) 𝐬𝐢𝐧 𝜽

𝒚′ = 𝒚𝒓 + (𝒙 − 𝒙𝒓) 𝐬𝐢𝐧 𝜽 + (𝒚 − 𝒚𝒓) 𝐜𝐨𝐬 𝜽

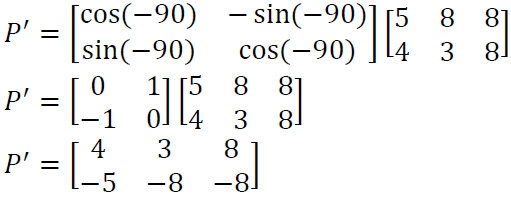
* These equations are different from rotation about origin and its matrix representation is also different.
* Rotation is also rigid body transformation so we need to rotate each point of the object.
* **Rotation about an arbitrary point:** If we want to rotate an object or point about an arbitrary point, first of all, we ***translate*** the point about which we want to rotate to the origin. Then ***rotate*** the point or object about the origin, and at the end, we again ***translate*** it to the original place. We get rotation about an arbitrary point.



**Example**: - Locate the new position of the triangle [A (5, 4), B (8, 3), C (8, 8)] after its rotation by 90o ***clockwise*** about the origin.

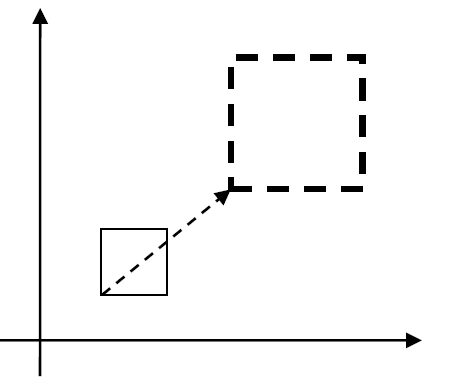
**Solution**-As rotation is clockwise we will take 𝜃 = −90°.

𝑃′ = 𝑅 ∙ 𝑃



Final coordinates after rotation are [A’ (4, -5), B’ (3, -8), C’ (8, -8)].

# Scaling



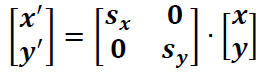
* It is a transformation that is used to alter the size of an object.
* This operation is carried out by multiplying coordinate value (𝒙, 𝒚) with scaling factor (𝒔𝒙, 𝒔𝒚) respectively.
* Equation for scaling is given by:

𝒙′ = 𝒙 ∙ 𝒔𝒙

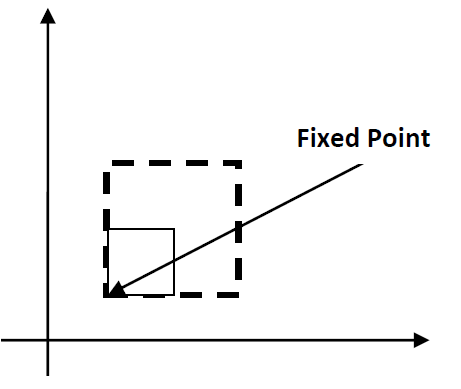
𝒚′ = 𝒚 ∙ 𝒔𝒚

* These equations can be represented in column vector matrix equation as:

𝑷′ = 𝑺 ∙ 𝑷



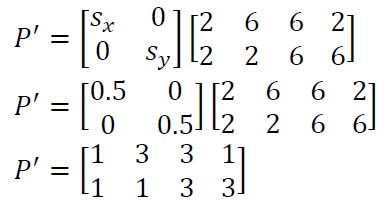
* Any positive value can be assigned to (𝒔𝒙, 𝒔𝒚).
* Values less than 1 reduce the size while values greater than 1 enlarge the size of the object, and the object remains unchanged when values of both factors are 1.
* Same values of 𝒔𝒙 and 𝒔𝒚 will produce Uniform Scaling. And different values of 𝒔𝒙 and 𝒔𝒚 will produce Differential Scaling.
* If the picture is enlarged to twice its original size, then Sx = Sy =2. If Sx and Sy are not equal, then scaling will occur but it will elongate or distort the picture.
* Objects transformed with the above equation are both scale and repositioned.
* Scaling factor with value less than 1 will move object closer to origin, while scaling factor with value greater than 1 will move object away from origin.
* We can control the position of object after scaling by keeping one position fixed called **Fix point** (𝒙𝒇, 𝒚𝒇) that point will remain unchanged after the scaling transformation.



**Example**: - Consider square with left-bottom corner at (2, 2) and right-top corner at (6, 6) apply the transformation which makes its size half.

**Solution**- As we want size half so value of scale factor is 𝑠𝑥 = 0.5, 𝑠𝑦 =0.5 and Coordinates of square are [A (2,2), B (6, 2), C (6, 6), D (2, 6)].

𝑃′ = 𝑆 ∙ 𝑃



Final coordinates after scaling are [A’ (1, 1), B’ (3, 1), C’ (3, 3), D’ (1, 3)].

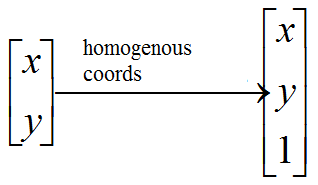
**------------------------------------------------------------------------**

**Matrix Representation and homogeneous coordinates**

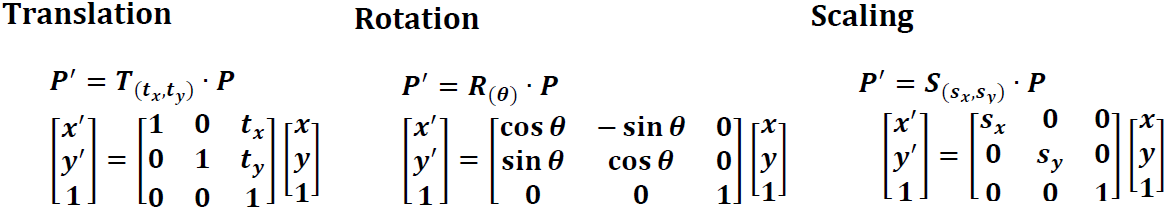
* Many graphics applications involve ***sequences*** of geometric transformations.
* For example, in design and picture construction application we perform Translation, Rotation, and scaling to fit the picture components into their proper positions.
* We have seen that each of the three basic two-dimensional transformations (translation, rotation, and scaling) can be expressed in the general matrix form

P′ = M1 · P +M2

* with coordinate positions P′ and P represented as column vectors. Matrix M1 is a 2 × 2 array containing multiplicative factors, and M2 is a two-element column matrix containing translational terms. For translation, M1 is the identity matrix. For rotation or scaling, M2 contains the translational terms associated with the pivot point or scaling fixed point. To produce a sequence of transformations with these equations, such as scaling followed by rotation and then translation, we could calculate the transformed coordinates one step at a time. First, coordinate positions are scaled, then these scaled coordinates are rotated, and finally, the rotated coordinates are translated. A more efficient approach, however, is to combine the transformations so that the final coordinate positions are obtained directly from the initial coordinates, without calculating intermediate coordinate values. We can do this by reformulating Equation to eliminate the matrix addition operation.
* For two dimensional geometric transformations we can take value of 𝒉 is any positive number so we can get infinite homogeneous representation for coordinate value (𝒙, 𝒚).
* But convenient choice is set 𝒉 = 𝟏 as it is multiplicative identity, then (𝒙, 𝒚) is represented as (𝒙, 𝒚, 𝟏).



* Expressing coordinates in homogeneous coordinates form allows us to represent all geometric transformation equations as matrix multiplication.
* Let’s see each representation with 𝒉 = 𝟏

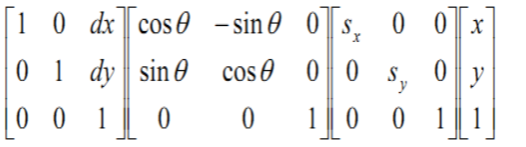


**Composite Transformation**

* We can set up a matrix for any sequence of transformations as a composite transformation matrix by calculating the matrix product of individual transformation.
* For column matrix representation of coordinate positions, we form composite transformations by multiplying matrices in order ***from right to left.***

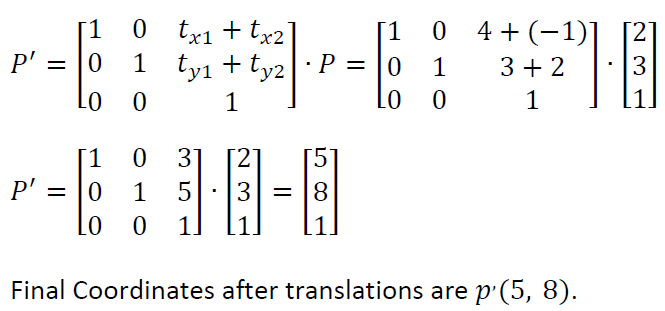
***Example:***

scales a point, then rotates it, then translates it:

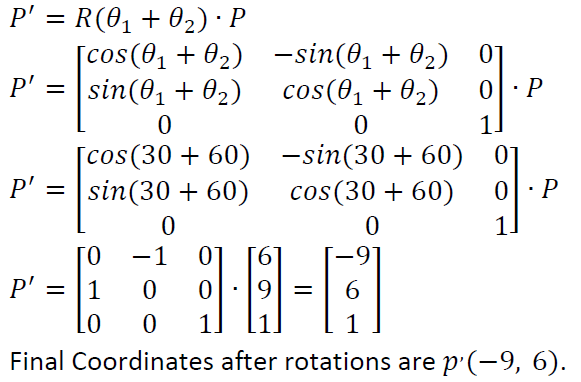


***Example*:** Obtain the final coordinates after two translations on point 𝑝 (2,3) with translation vector(4, 3) and (−1, 2) respectively.

𝑃′ = 𝑇 (𝑡𝑥1 + 𝑡𝑥2, 𝑡𝑦1 + 𝑡𝑦2) ∙ 𝑃

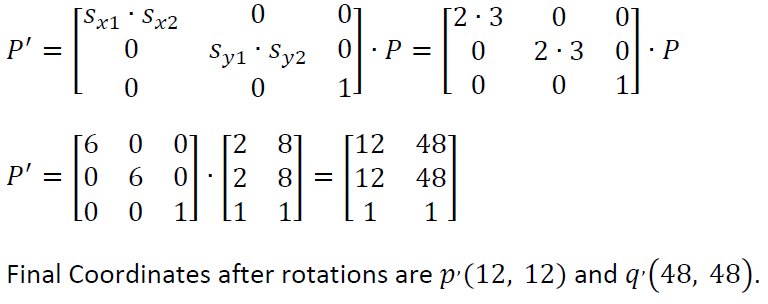


***Example*:** Obtain the final coordinates after two rotations on point 𝑝 (6,9) with rotation angles are 30𝑜 and 60𝑜 respectively.



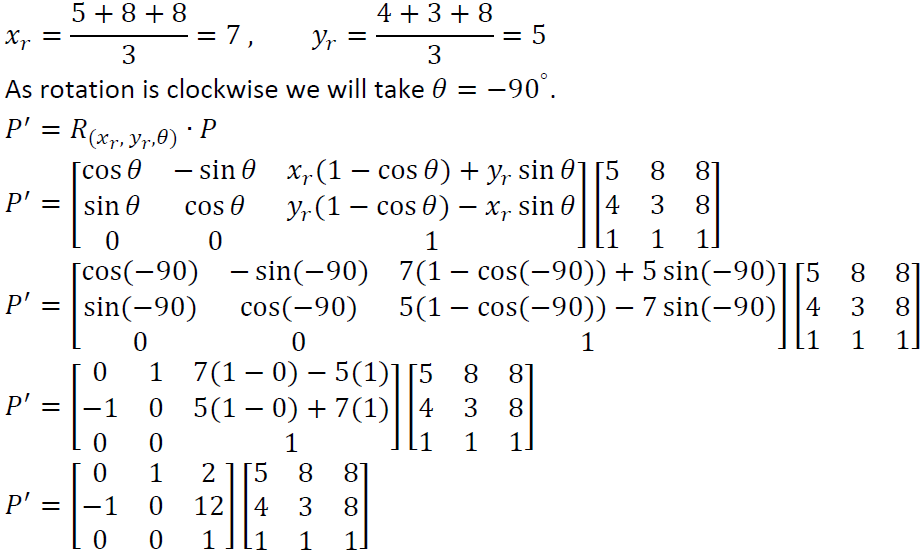
***Example*:** Obtain the final coordinates after two scaling on line 𝑝𝑞 [𝑝 (2,2), 𝑞 (8, 8)] with scaling factors are (2, 2) and (3, 3) respectively.

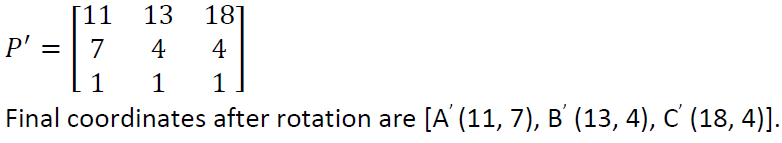
𝑃′ = 𝑆 (𝑠𝑥1 ∙ 𝑠𝑥2, 𝑠𝑦1 ∙ 𝑠𝑦2) ∙ 𝑃



***Example***: - Locate the new position of the triangle [A (5, 4), B (8, 3), C (8, 8)] after its rotation by 90o clockwise about the centroid of the triangle.

Pivot point is centroid of the triangle so:





**------------------------------------------------------------**

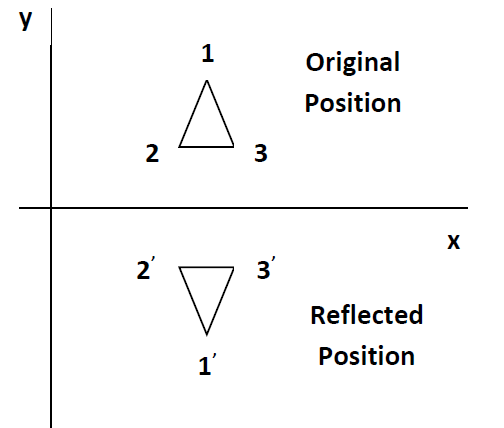
**Other Transformation**

Some packages provide few additional transformations which are useful in certain applications. Two such transformations are reflection and shearing

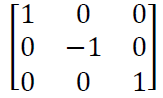
# Reflection

* A reflection is a transformation that produces a mirror image of an object.
* The mirror image for a two –dimensional reflection is generated relative to an **axis of reflection** by rotating the object 180o about the reflection axis.
* Reflection gives an image based on the position of the axis of reflection. Transformation matrices for a few positions are discussed here.

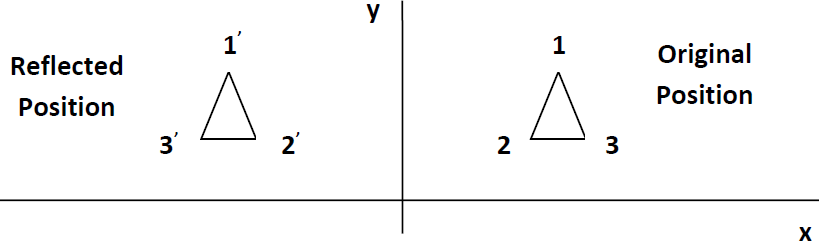
1. **Transformation matrix for reflection about the line 𝒚 = 𝟎, 𝒕𝒉𝒆 𝒙 𝒂𝒙𝒊𝒔.**



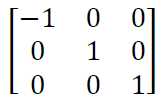
* This transformation keeps x values the same, but flips (Change the sign) y values of coordinate positions.



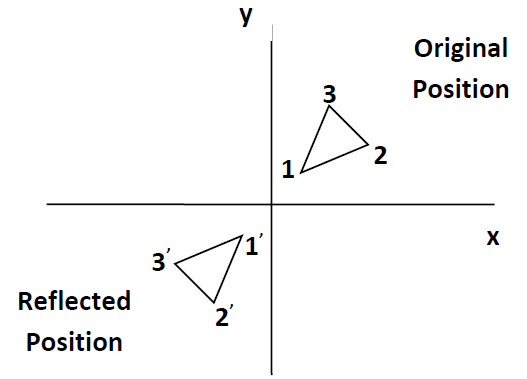
1. **Transformation matrix for reflection about the line 𝒙 = 𝟎 , 𝒕𝒉𝒆 𝒚 𝒂𝒙𝒊𝒔.**



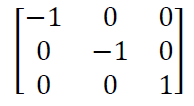
* This transformation keeps y values the same, but flips (Change the sign) x values of coordinate positions.



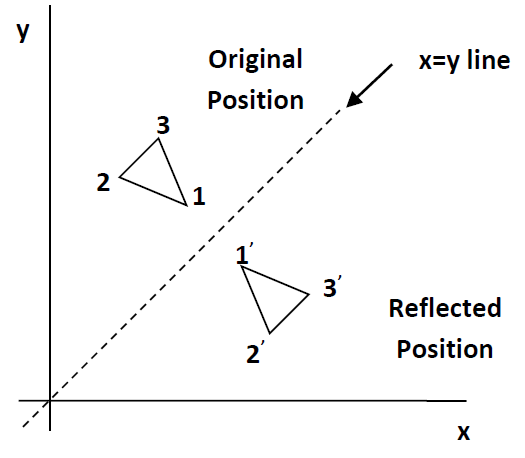
1. **Transformation matrix for reflection about the 𝑶𝒓𝒊𝒈𝒊𝒏.**



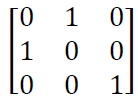
* This transformation flips (Change the sign) x and y both values of coordinate positions.



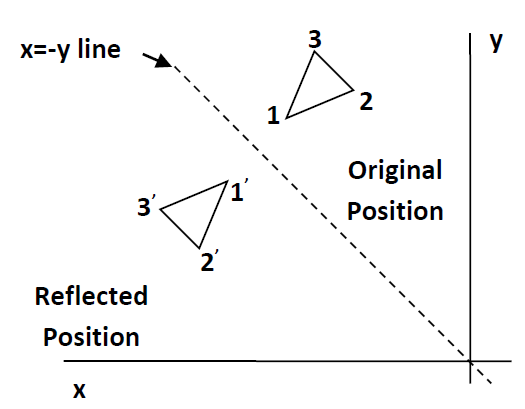
1. **Transformation matrix for reflection about the line 𝒙 = 𝒚.**



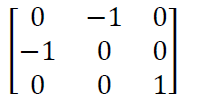
* This transformation interchanges x and y values of coordinate positions.



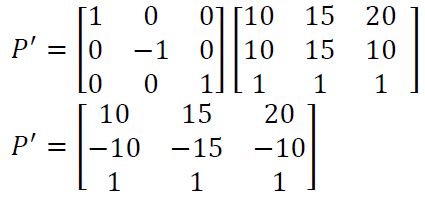
1. **Transformation matrix for reflection about the line 𝒙 = −𝒚.**



* This transformation interchanges x and y values of coordinate positions.



**Example**: - Find the coordinates after reflection of the triangle [A (10, 10), B (15, 15), C (20, 10)] about x axis.

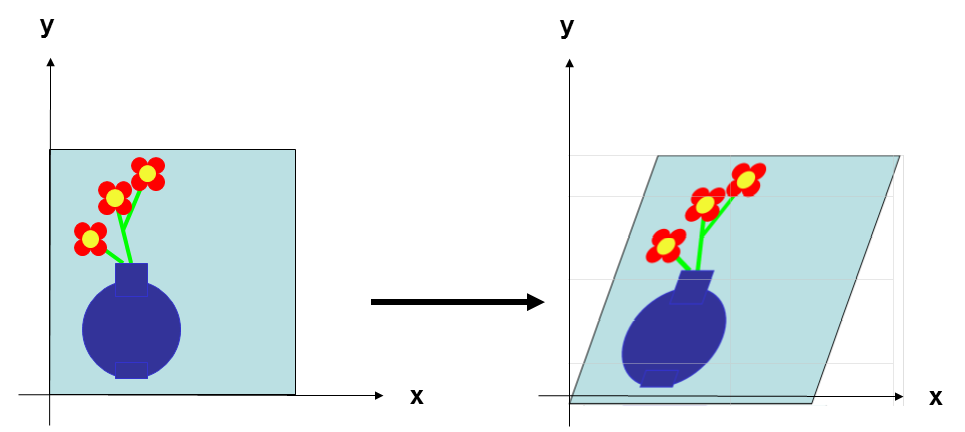


Final coordinate after reflection are [A’ (10, -10), B’ (15, -15), C’ (20, -10)]

# Shearing:

* It is transformation which changes the shape of an object. The sliding of layers of objects occurs. The shear can be in one direction or in two directions.
* The shape of the object is distorted by producing the sliding effect of layers over each other.

***EX***. Shear relative to x

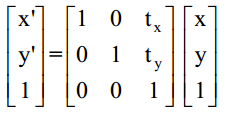


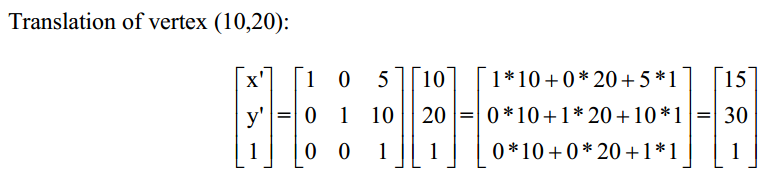
* **The transformation matrix:**

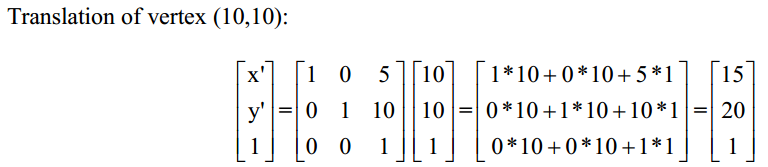


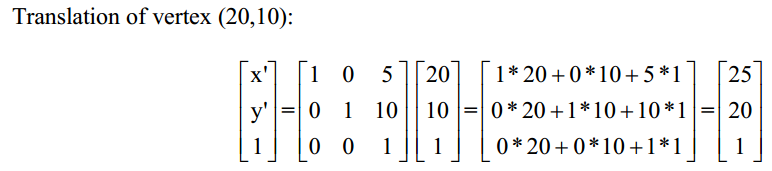
* Where hx is a shear parameter which can take any real number value.
* **Q1:** Translate a triangle with vertices at original coordinates (10,20), (10,10), (20,10) by tx=5, ty=10

**Solution:**



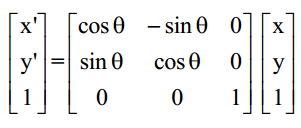


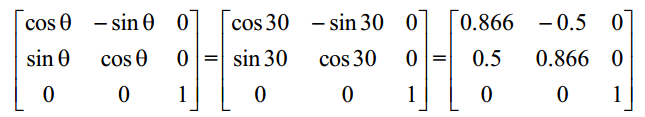


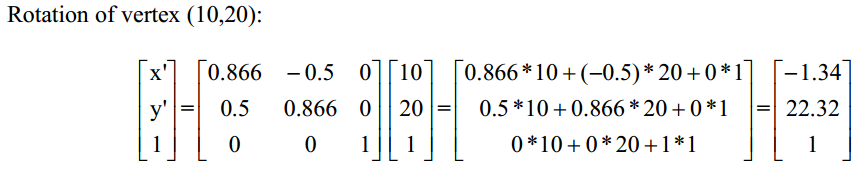


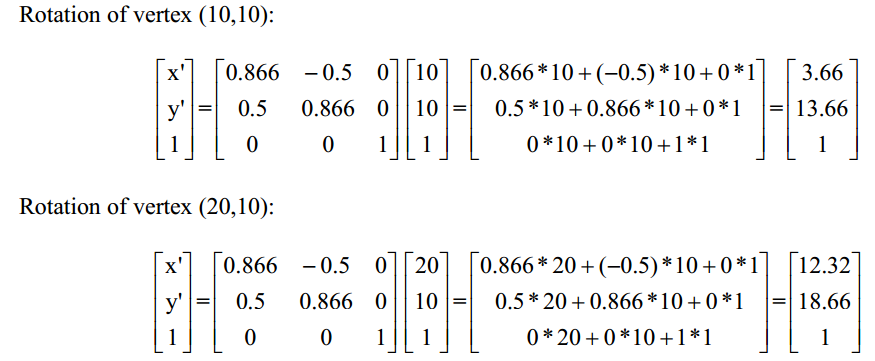
* The resultant coordinates of the triangle vertices are (15,30), (15,20), and (25,20) respectively.
* **Q2:** Rotate a triangle about the origin with vertices at original coordinates (10,20), (10,10), (20,10) by 30 degrees

**Solution**





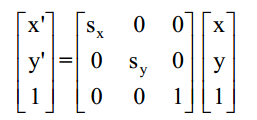


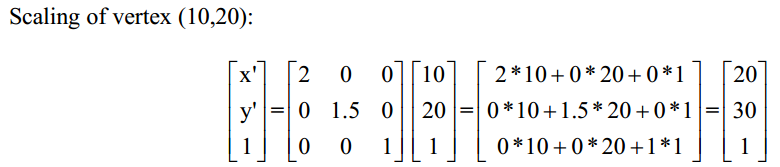


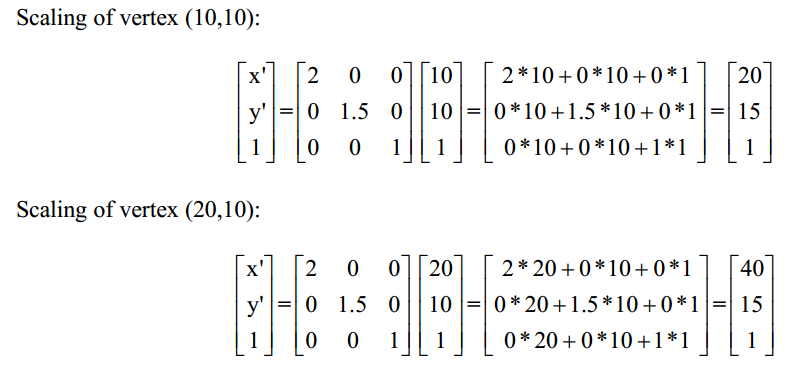
The resultant coordinates of the triangle vertices are (-1.34,22.32), (3.6,13.66), and (12.32,18.66) respectively.

* **Q3:** Scale a triangle with respect to the origin, with vertices at original coordinates (10,20), (10,10), (20,10) by sx=2, sy=1.5

**Solution:**



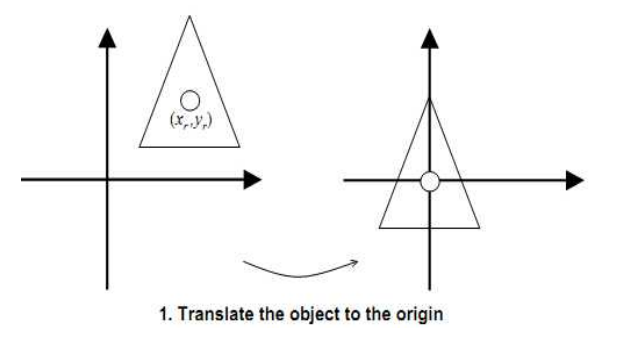


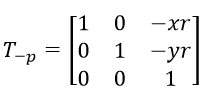


The resultant coordinates of the triangle vertices are (20,30), (20,15), and (40,15) respectively.

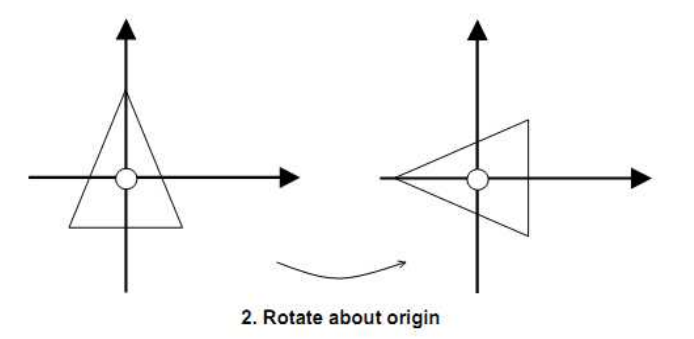
* **Rotation About a Fixed Point**
* It is often required to rotate an object around an arbitrary point rather than the origin.
* Suppose we want to rotate an object around an arbitrary point **(xr, yr)**
* **Steps:**

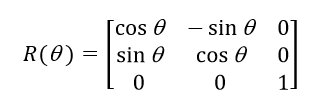
1. Translate the object to the origin.  
2. Rotate around the origin.  
3. Translate the object back.



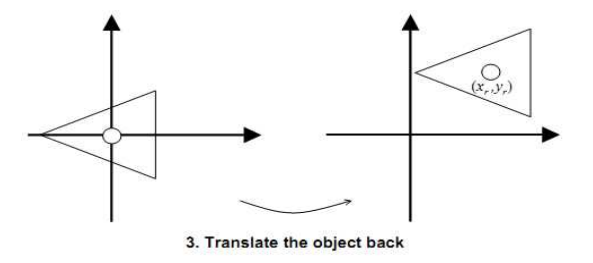


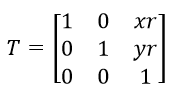
* Now the center of rotation is at the origin, we can apply our rotation matrix by degree θ



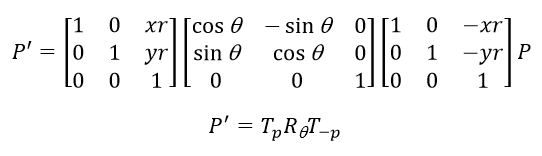


* We translate the object back to its original position:





* **So final compound matrix is:**



1. <https://www.javatpoint.com/computer-graphics-introduction-of-transformations> [↑](#footnote-ref-0)