Chapter 4

Two Dimensional Transformation

In many cases a complex picture can always be treated as a combination of straight line, circles, ellipse etc., and if we are able to generate these basic figures, we can also generate combinations of them. Once we have drawn these pictures, the need arises to transform these pictures.

Transformation means changing some graphics into something else by applying rules. We can have various types of transformations such as translation, scaling up or down, rotation, shearing, etc. When a transformation takes place on a 2D plane, it is called **2D transformation.**

Transformations play an important role in computer graphics to reposition the graphics on the screen and change their size or orientation.

The three basic transformations are

a) Translation

b) Rotation

c) Scaling.

Other transformation includes reflection and shear.

For each cases of transformation, we consider the reference / pivot point is origin, so if we have to do these transformations for a point P(x, y) about any arbitrary point (x_r, y_r), then we have to first shift the given point to the origin and then perform required transformation and finally shift to that arbitrary point position.

Geometric transformations involve taking a **preimage** and transforming it in some way to produce an **image**.

There are three basic classes of transformations:

1. **Rigid body**
   - Preserves distance and angles.
   - Examples: translation and rotation.
2. **Conformal**
   - Preserves angles.
   - Examples: translation, rotation, and uniform scaling.
3. **Affine**
   - Preserves parallelism. Lines remain lines.
   - Examples: translation, rotation, scaling, shear, and reflection

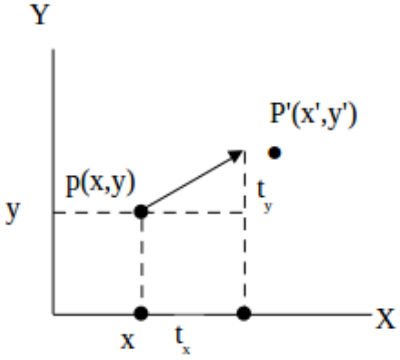
a) Translation

Repositioning of object along a straight-line path from one coordinate location to another is called translation. We translate a two-dimensional point by adding translation distances t_x , and t_y , to the original coordinate position (x, y) to move the point to a new position (x ' , y') as:

$$x' = x + t_x$$

$$y' = x + t_y$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$



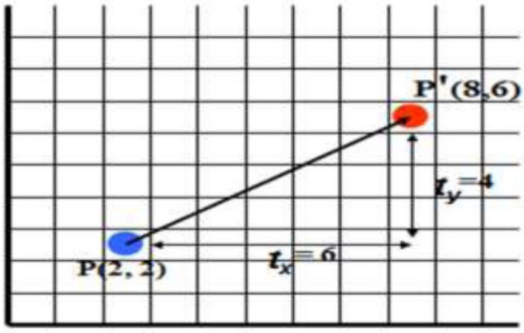
The translation distance pair (t_x,t_y) is known as translation vector or

shift vector. We can express translation equations as matrix representations as

$$P=\begin{bmatrix} x \\ y \end{bmatrix} \qquad P'=\begin{bmatrix} x' \\ y' \end{bmatrix} \quad T=\begin{bmatrix} t_x \\ t_y \end{bmatrix} \qquad \therefore P'=P+T$$

For Example:

Given a point P with original position (2,2). Then after performing translation operation with t_x = 6 and t_y = 4, we get new transformed coordinate P' with coordinate (8,6).



b) Rotation

A two-dimensional rotation is applied to an object by repositioning it along a circular path in the xy plane. To generate a rotation, we specify a rotation angle 'θ' and the position (x_r , y_r ,) of the rotation point (or pivot point) about which the object is to be rotated.

- + Value for 'θ' define counter-clockwise rotation about a point
- - Value for 'θ' defines clockwise rotation about a point

Let (x, y) is the original point, 'r' the constant distance from origin, & 'Φ' the original angular displacement from x-axis. Now the point (x, y) is rotated through angle 'θ' in a counter clock wise direction

we can express the transformed coordinates in terms of 'Φ' and 'θ' as

x' = r cos(Φ + θ) = r cosΦ . cosθ - r sin Φ. sinθ ....(i)

y' = r sin(Φ + θ) = r cosΦ . sinθ + r sinΦ . cosθ ....(ii)

We know that original coordinates of point in polar coordinates are

x = r cos Φ

y = r sin Φ

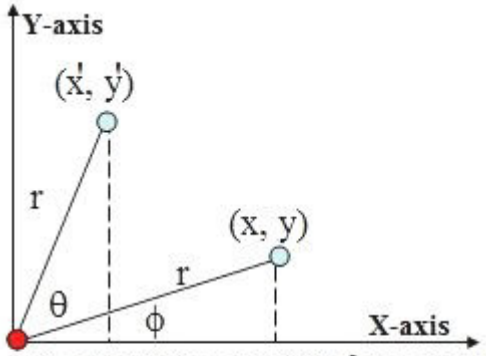


Fig. Rotating a point from position (x, y) to position (x', y ') through an angle θ about rotation point (0, 0). The original angular displacement of the point from the x axis is ϕ.

Substituting these values in (i) and (ii), we get,

x' = x cosθ - y sinθ

y' = x sinθ + y cosθ

So using column vector representation for coordinate points the matrix form would be P' = R . P

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

Rotation of a point about an arbitrary pivot position

- Translate the point (x_r,y_r) and P(x,y) by translation vector $(-x_r, -y_r)$ which translates the pivot to origin and P(x,y) to $(x-x_r, y-y_r)$.
- Now apply the rotation equations when pivot is at origin to rotate the translated point $(x-x_r, y-y_r)$ as:

$$x_1 = (x - x_r)\cos\theta - (y - y_r)\sin\theta$$

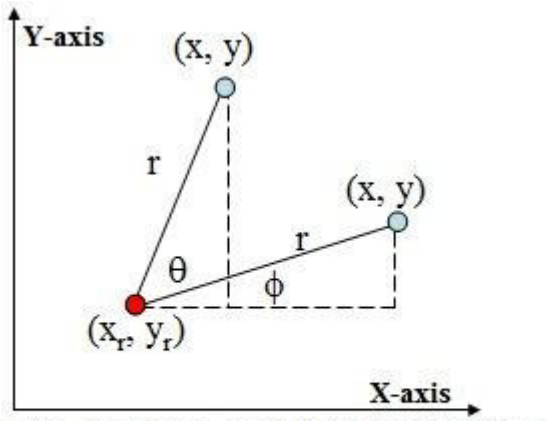
$$y_1 = (x - x_r)\sin\theta + (y - y_r)\cos\theta$$

- Re- translate the rotated point (x_1, y_1) with translation vector (x_r,y_r) which is reverse translation to original translation.

Finally we get the equation after successive transformation as

$$x' = x_r + (x - x_r)\cos\theta - (y - y_r)\sin\theta \quad \ldots\ldots\ldots\ldots 1$$

$$y' = y_r + (x - x_r)\sin\theta + (y - y_r)\cos\theta \quad \ldots\ldots\ldots\ldots 2$$

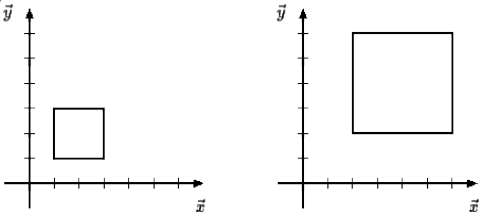


Which are actually the equations for rotation of (x,y) from the pivot point (x_r,y_r)

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 1-\cos\theta & -\sin\theta \\ -\sin\theta & 1-\cos\theta \end{pmatrix} \begin{pmatrix} x_r \\ y_r \end{pmatrix}$$

c) Scaling

A scaling transformation alters the size of an object. This operation can be carried out for polygons by multiplying the coordinate values (x, y) of each vertex by scaling factors s_x and s_y to produce the transformed coordinates (x', y').



- s_x scales object in 'x' direction
- s_y scales object in 'y' direction

Thus, for equation form,

x' = x. s_x and y' = y. s_y

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} S_x & 0 \\ 0 & S_y \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

- Values greater than 1 for s_x s_y produce enlargement
- Values less than 1 for s_x s_y reduce size of object
- $s_x = s_y = 1$ leaves the size of the object unchanged
- When s_x , s_y are assigned the same value $s_x = s_y = 4$ then a Uniform Scaling is produced.

- Scaling **About arbitrary point**

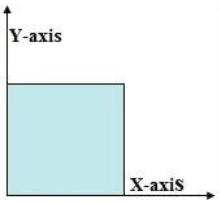
If p(x,y) be scaled to a point p'(x', y') by s_x time in X-units and s_y times in y-units about arbitrary point (x_f, y_f) then the equation for scaling is given as

$$x' = x_f + s_x .(x - x_f)$$

$$y' = y_f + s_y .(y - y_f)$$

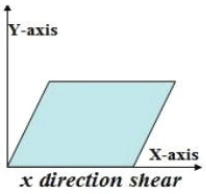
d. Shearing

A transformation that distorts the shape of an object such that the transformed shape appears as if the object were composed of internal layers that had been caused to slide over each other is called shear.



X-direction Shear:

An X-direction shear relative to x-axis is produced with transformation matrix equation.



In 'x' direction,

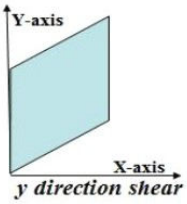
$x' = x + s_{hx} . y$

$y' = y$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & s_{hx} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Y-direction Shear:

An y-direction shear relative to y-axis is produced by following transformation equations.



In 'y' direction,

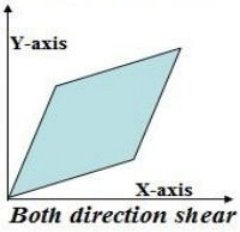
$x' = x$

$y' = y + s_{hy} . x$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ s_{hy} & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Both direction Shear:

Both direction share relative to x-axis and y-axis is produced by following transformation equations

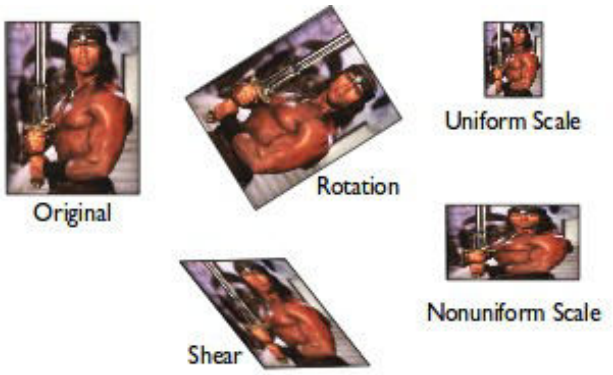


In both directions,

$x' = x + s_{hx} . y$

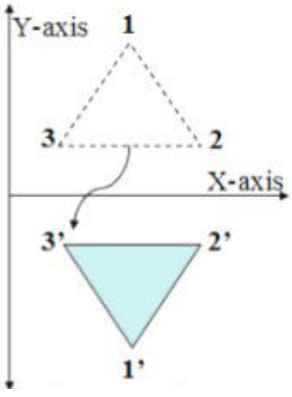
$y' = y + s_{hy} . x$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & s_{hx} \\ s_{hy} & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$



e. Reflection

A reflection is a transformation that produce a mirror image of an object. In 2D-transformation, reflection is generated relative to an axis of reflection. The reflection of an object to an relative axis of reflection , is same as 180° rotation about the reflection axis.



i. **Reflection about x axis or about line y = 0**

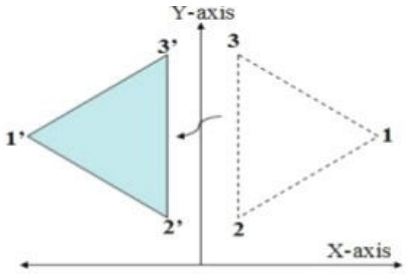
Keeps 'x' value same but flips y value of coordinate points

x' = x
y'= - y

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

ii. **Reflection about y axis or about line x = 0**

Keeps 'x' value same but flips y value of coordinate points

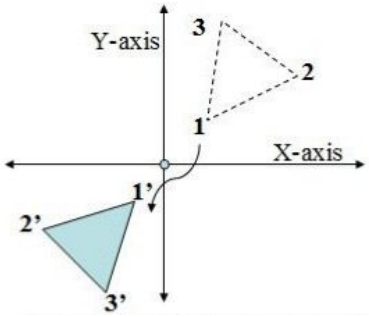


x' = - x
y'= y

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

iii. **Reflection about origin**

Flip both 'x' and 'y' coordinates of a point



x' = - x
y' = -y

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Q. What is Arbitrary axis?

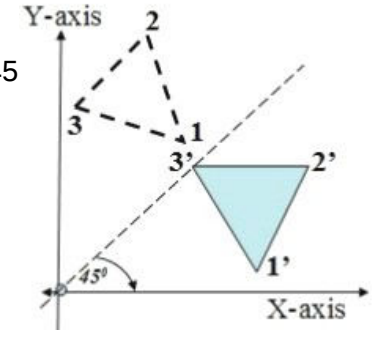
Arbitrary axis simply means the axis chosen in any way we like within the co-ordinate system. We choose any line in space, and then put a pin into our object along that line, and transform the object around the pin

iv. **Reflection on an arbitrary axis**

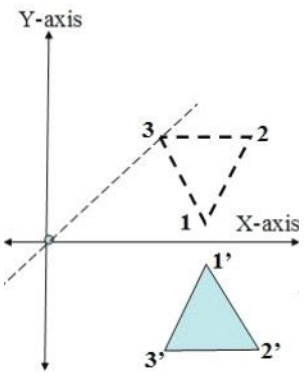
The reflection on any arbitrary axis of reflection can be achieved by sequence of rotation and co-ordinate axes reflection matrices.

Reflection about line y = x (θ=45^0)

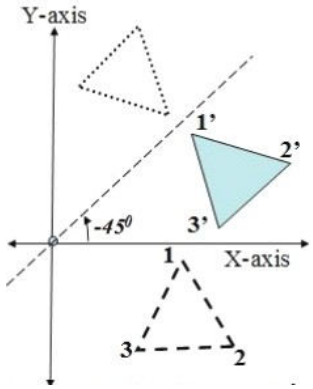
- Rotate about origin in clockwise direction by 45 degree which rotates line y = x to x-axis



- Take reflection against x-axis



- Rotate in anti-clockwise direction by same angle



Here, we have multiple transformation at once, So when more than one transformations are applied for performing a task then such transformation is called **composite transformation.**

So the composite transformation required for reflecting the given object about y=x axis is

$$T = R_{\theta=45} R_x R_{\theta=-45}$$

Forming products of transformation matrices is often referred to as a concatenation, or composition, of matrices.

Q. What is the basic purpose of composite transformation?

The basic purpose of composing transformations is to gain efficiency by applying a single composed transformation to a point, rather than applying a series of transformation, one after another.

Q. Translate the given points (2,5) by the translating value (3,3).

Solution:

Given Point P (2, 5) and translation distance Tx= 3, and Ty= 3
We have From Translation Matrix

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

i.e.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 2 \\ 5 \end{bmatrix} + \begin{bmatrix} 3 \\ 3 \end{bmatrix} = \begin{bmatrix} 5 \\ 8 \end{bmatrix}$$

Therefore, P(2,5) is translated to new point P'(5,8).

Q. Translate the given square having coordinate A (0,0), B (3,0), and C (3,3) and D (0, 3) by the translating value 2 in both directions.

Solution:

Given points A (0,0), B (3,0), and C (3,3) and D (0, 3) and translating value 2 (t_x=t_y=2)

We have From Translation Matrix

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

i.e.

A'= $\begin{bmatrix} 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 2 \\ 2 \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \end{bmatrix}$

B'= $\begin{bmatrix} 3 \\ 0 \end{bmatrix} + \begin{bmatrix} 2 \\ 2 \end{bmatrix} = \begin{bmatrix} 5 \\ 2 \end{bmatrix}$

C'= $\begin{bmatrix} 3 \\ 3 \end{bmatrix} + \begin{bmatrix} 2 \\ 2 \end{bmatrix} = \begin{bmatrix} 5 \\ 5 \end{bmatrix}$

D'= $\begin{bmatrix} 0 \\ 3 \end{bmatrix} + \begin{bmatrix} 2 \\ 2 \end{bmatrix} = \begin{bmatrix} 2 \\ 5 \end{bmatrix}$

Hence the final co-ordinate are A'(2,2), B'(5,2), C'(5,5) and D'(2,5)

Homogeneous Co-ordinates

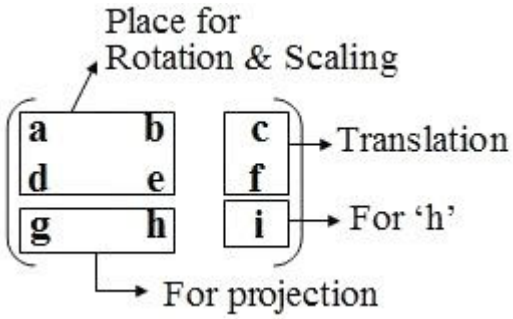
To perform a sequence of transformation such as translation followed by rotation and scaling, we need to follow a sequential process −

- Translate the coordinates,
- Rotate the translated coordinates, and then
- Scale the rotated coordinates to complete the composite transformation.

To shorten this process, we have to use 3×3 transformation matrix instead of 2×2 transformation matrix. To convert a 2×2 matrix to 3×3 matrix, we have to add an extra **dummy coordinate**. So, we can represent the point by 3 numbers instead of 2 numbers, which is called Homogenous Coordinate system.

In this system, we can represent all the transformation equations in matrix multiplication.

Any Cartesian point P(X, Y) can be converted to homogenous coordinates by P' (X_h, Y_h, h). The 'h' is normally set to 1. If the value of 'h' is more the one value then all the co-ordinate values are scaled by this value.



Coordinates of a point are represented as three element column vectors, transformation operations are written as 3 x 3 matrices.

For translation

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

With T(t_x , t_y) as translation matrix, inverse of this translation matrix is obtained by

representing t_x , t_y with $-t_x$, $-t_y$.

For rotation

a. Counter Clock Wise (CCW):

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} \text{Cos}\theta & -\text{Sin}\,\theta & 0 \\ \text{Sin}\theta & \text{Cos}\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

b. Clock Wise(CCW):

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} \text{Cos}\theta & \text{Sin}\,\theta & 0 \\ -\text{Sin}\theta & \text{Cos}\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

For scaling

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} S_{hx} & 0 & 0 \\ 0 & S_{hy} & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Inverse scaling matrix is obtained with $1 / s_{hx}$ and $1 / s_{hy}$.

For Reflection

a) Reflection about x-axis

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

b) Reflection about y-axis

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

c) Reflection about y=x-axis

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

d) Reflection about y=-x-axis

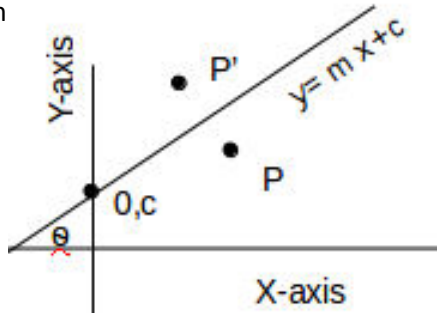
$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

e) Reflection about any line y=mx+c

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{-(m^2-1)}{(m^2+1)} & \frac{2m}{(m^2+1)} & \frac{2mc}{(m^2+1)} \\ \frac{2m}{(m^2+1)} & \frac{(m^2-1)}{(m^2+1)} & \frac{2c}{(m^2+1)} \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Q. Derive the composite matrix for reflecting an object about any arbitrary line y=mx+c.

In order to reflect an object about any line y=mx+c, we need to perform composite transformation as below



$$T = T_{(0,c)} \cdot R_{\Theta} \cdot R_{x} \cdot R_{-\Theta} \cdot T_{(0,-c)}$$

And

Slope m= tan Θ,

Also we have,

$$\cos^2\Theta = \frac{1}{\tan^2\theta+1} = \frac{1}{m^2+1} = \frac{1}{\sqrt{m^2+1}}$$

Also we have,

$$\sin^2\Theta + \cos^2\Theta = 1$$

$$\sin^2\Theta = 1 - \cos^2\Theta$$

$$\sin^2\Theta = 1 - \frac{1}{m^2+1} = \frac{m^2+1-1}{m^2+1}$$

$$\sin\Theta = \frac{m}{\sqrt{m^2+1}}$$

So,

$$T = T_{(0,c)} \cdot R_{\Theta} \cdot R_{x} \cdot R_{-\Theta} \cdot T_{(0,-c)}$$

$$= \begin{bmatrix}1&0&0\\0&1&c\\0&0&1\end{bmatrix} \begin{bmatrix}\cos\theta&-\sin\theta&0\\\sin\theta&\cos\theta&0\\0&0&1\end{bmatrix} \begin{bmatrix}1&0&0\\0&-1&0\\0&0&1\end{bmatrix} \begin{bmatrix}\cos\theta&\sin\theta&0\\-\sin\theta&\cos\theta&0\\0&0&1\end{bmatrix} \begin{bmatrix}1&0&0\\0&1&-c\\0&0&1\end{bmatrix}$$

$$= \begin{bmatrix}1&0&0\\0&1&c\\0&0&1\end{bmatrix} \begin{bmatrix}\cos\theta&-\sin\theta&0\\\sin\theta&\cos\theta&0\\0&0&1\end{bmatrix} \begin{bmatrix}1&0&0\\0&-1&0\\0&0&1\end{bmatrix} \begin{bmatrix}\cos\theta&\sin\theta&-c\sin\theta\\-\sin\theta&\cos\theta&-c\cos\theta\\0&0&1\end{bmatrix}$$

$$= \begin{bmatrix}1&0&0\\0&1&c\\0&0&1\end{bmatrix} \begin{bmatrix}\cos\theta&-\sin\theta&0\\\sin\theta&\cos\theta&0\\0&0&1\end{bmatrix} \begin{bmatrix}\cos\theta&\sin\theta&-c\sin\theta\\\sin\theta&-\cos\theta&c\cos\theta\\0&0&1\end{bmatrix}$$

$$= \begin{bmatrix}1&0&0\\0&1&c\\0&0&1\end{bmatrix} \begin{bmatrix}\frac{1}{\sqrt{(m^2+1)}}&\frac{-m}{\sqrt{(m^2+1)}}&0\\\frac{-m}{\sqrt{(m^2+1)}}&\frac{1}{\sqrt{(m^2+1)}}&0\\0&0&1\end{bmatrix} \begin{bmatrix}\frac{1}{\sqrt{(m^2+1)}}&\frac{m}{\sqrt{(m^2+1)}}&\frac{-cm}{\sqrt{(m^2+1)}}\\\frac{m}{\sqrt{(m^2+1)}}&\frac{-1}{\sqrt{(m^2+1)}}&\frac{c}{\sqrt{(m^2+1)}}\\0&0&1\end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & c \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{1-m^2}{m^2+1} & \frac{2m}{m^2+1} & \frac{-2cm}{m^2+1} \\ \frac{2m}{m^2+1} & \frac{m^2-1}{m^2+1} & \frac{c-cm^2}{m^2+1} \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \frac{1-m^2}{m^2+1} & \frac{2m}{m^2+1} & \frac{-2mc}{m^2+1} \\ \frac{2m}{m^2+1} & \frac{m^2-1}{m^2+1} & \frac{2c}{m^2+1} \\ 0 & 0 & 1 \end{bmatrix}$$

Q. Rotate a triangle A(0,0) , B(2,2) , C(4,2) about the origin by the angle of 45 degree.

Solution:
The given triangle ABC can be represented by a matrix, formed from the homogeneous coordinates of the vertices.

$$\begin{bmatrix} 0 & 2 & 4 \\ 0 & 2 & 2 \\ 1 & 1 & 1 \end{bmatrix}$$

Also, we have

$$R_{45^0} = \begin{bmatrix} cos45^0 & -sin45^0 & 0 \\ sin45^0 & cos45^0 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} & 0 \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

So the coordinates of the rotated triangle ABC are

$$R_{45^0}[ABC] = \begin{bmatrix} \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} & 0 \\ \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 & 2 & 4 \\ 0 & 2 & 2 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & \sqrt{2} \\ 0 & 2\sqrt{2} & 3\sqrt{2} \\ 1 & 1 & 1 \end{bmatrix}$$

Hence the final co-ordinate points are A'(0,0), B'(0, 2√2) and C'(√2, 3√2).

Q. Rotate the triangle (5, 5), (7, 3), (3, 3) about fixed point (5, 4) in counter clockwise (CCW) by 90 degree.

Solution

The required steps are:

1. Translate the fixed point to origin.
2. Rotate about the origin by 90 degree.
3. Reverse the translation as performed earlier.

Thus, the composite matrix is given by

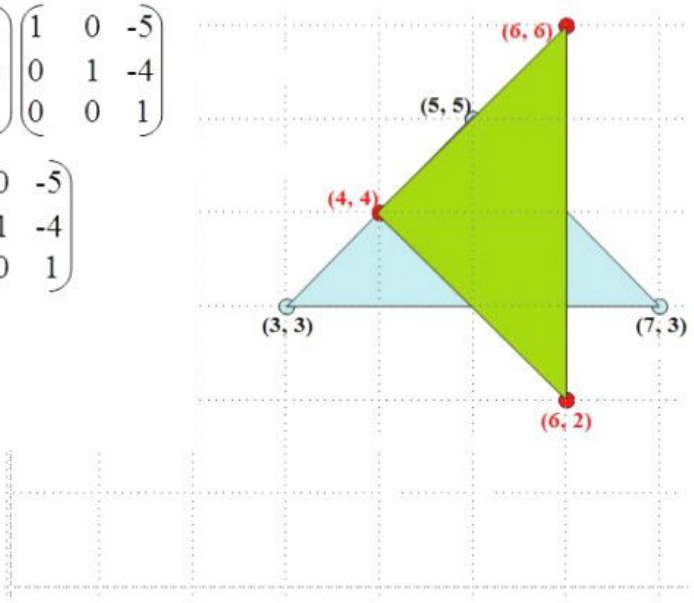
$$Com = T_{(xf, yf)} R_{\theta} T_{(-xf, -yf)}$$

$$= \begin{pmatrix} 1 & 0 & 5 \\ 0 & 1 & 4 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} Cos90 & -Sin90 & 0 \\ Sin90 & Cos90 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & -5 \\ 0 & 1 & -4 \\ 0 & 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 0 & 5 \\ 0 & 1 & 4 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & -5 \\ 0 & 1 & -4 \\ 0 & 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 0 & 5 \\ 0 & 1 & 4 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & -1 & 4 \\ 1 & 0 & -5 \\ 0 & 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} 0 & -1 & 9 \\ 1 & 0 & -1 \\ 0 & 0 & 1 \end{pmatrix}$$



Now, the required co-ordinate can be calculated as:

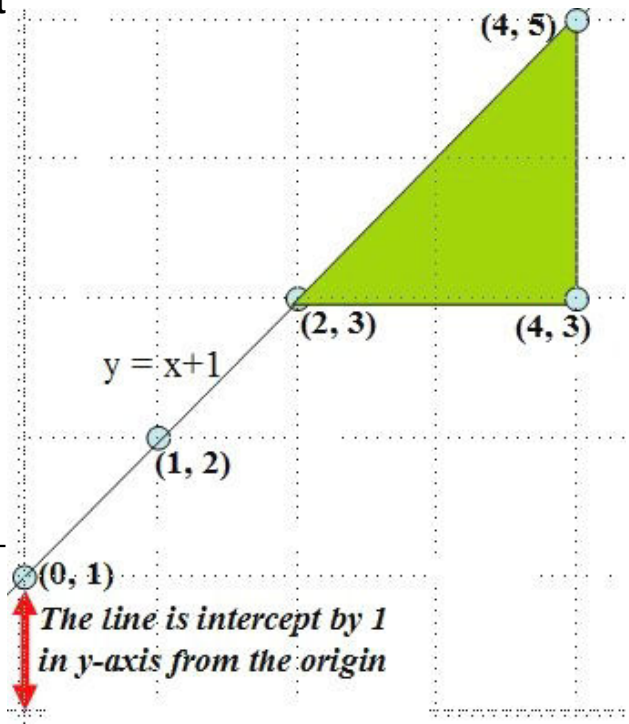
P' = Com * P

$$= \begin{pmatrix} 0 & -1 & 9 \\ 1 & 0 & -1 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 5 & 7 & 3 \\ 5 & 3 & 3 \\ 1 & 1 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} 4 & 6 & 6 \\ 4 & 6 & 2 \\ 1 & 1 & 1 \end{pmatrix}$$

Hence, the new required coordinate points are (4, 4), (6, 6) & (6, 2).

Q. Reflect an object (2, 3), (4, 3), (4, 5) about line y = x +1.



Here,
The given line is y = x +1.
Thus,

When x = 0, y=1
When x = 1, y=2
When x = 2, y=3

Also,
The slope of the line (m) = 1
Thus, the rotation angle (θ) = $Tan^{-1}(m) = Tan^{-1}(1)=45^0$
Here, the required steps are:

1. Translate the line to origin by decreasing the y-intercept with one.

2. Rotate the line by angle 45 0 in clockwise direction so that the given line must overlap x-axis.
3. Reflect the object about the x-axis.
4. Reverse rotate the line by angle -45 0 in counter-clockwise direction.
5. Reverse translate the line to original position by adding the y-intercept with one

Thus, the composite matrix is given by:

$$Com = \mathbf{T' R_\theta' R_x R_\theta T}$$

Thus, composite matrix is given by

$$= \underset{\textit{Addition y-intercept}}{\begin{pmatrix}1&0&0\\0&1&1\\0&0&1\end{pmatrix}} \underset{\textit{CCW Rotation}}{\begin{pmatrix}Cos45&-Sin45&0\\Sin45&Cos45&0\\0&0&1\end{pmatrix}} \underset{\textit{Reflection about x-axis}}{\begin{pmatrix}1&0&0\\0&-1&0\\0&0&1\end{pmatrix}} \underset{\textit{CW Rotation}}{\begin{pmatrix}Cos45&Sin45&0\\-Sin45&Cos45&0\\0&0&1\end{pmatrix}} \underset{\textit{Reduce y-intercept}}{\begin{pmatrix}1&0&0\\0&1&-1\\0&0&1\end{pmatrix}}$$

$$= \begin{pmatrix}1&0&0\\0&1&1\\0&0&1\end{pmatrix}\begin{pmatrix}1/\sqrt{2}&-1/\sqrt{2}&0\\1/\sqrt{2}&1/\sqrt{2}&0\\0&0&1\end{pmatrix}\begin{pmatrix}1&0&0\\0&-1&0\\0&0&1\end{pmatrix}\begin{pmatrix}1/\sqrt{2}&1/\sqrt{2}&0\\-1/\sqrt{2}&1/\sqrt{2}&0\\0&0&1\end{pmatrix}\begin{pmatrix}1&0&0\\0&1&-1\\0&0&1\end{pmatrix}$$

$$= \begin{pmatrix}1&0&0\\0&1&1\\0&0&1\end{pmatrix}\begin{pmatrix}1/\sqrt{2}&-1/\sqrt{2}&0\\1/\sqrt{2}&1/\sqrt{2}&0\\0&0&1\end{pmatrix}\begin{pmatrix}1&0&0\\0&-1&0\\0&0&1\end{pmatrix}\begin{pmatrix}1/\sqrt{2}&1/\sqrt{2}&-1/\sqrt{2}\\-1/\sqrt{2}&1/\sqrt{2}&-1/\sqrt{2}\\0&0&1\end{pmatrix}$$

$$= \begin{pmatrix}1&0&0\\0&1&1\\0&0&1\end{pmatrix}\begin{pmatrix}1/\sqrt{2}&-1/\sqrt{2}&0\\1/\sqrt{2}&1/\sqrt{2}&0\\0&0&1\end{pmatrix}\begin{pmatrix}1/\sqrt{2}&1/\sqrt{2}&-1/\sqrt{2}\\1/\sqrt{2}&-1/\sqrt{2}&1/\sqrt{2}\\0&0&1\end{pmatrix}$$

$$= \begin{pmatrix}1&0&0\\0&1&1\\0&0&1\end{pmatrix}\begin{pmatrix}0&1&-1\\1&0&0\\0&0&1\end{pmatrix}$$

$$= \begin{pmatrix}0&1&-1\\1&0&1\\0&0&1\end{pmatrix}$$

Now, the required co-ordinate can be calculated as:

$$P' = Com * P$$

$$= \begin{pmatrix}0&1&-1\\1&0&1\\0&0&1\end{pmatrix}\begin{pmatrix}2&4&4\\3&3&5\\1&1&1\end{pmatrix}$$

$$= \begin{pmatrix}2&2&4\\3&5&5\\1&1&1\end{pmatrix}$$

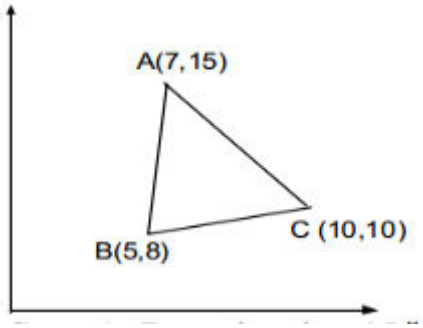
Hence, the final coordinates are (2, 3), (2, 5) & (4, 5)

Note: Composite Matrix can also be calculated as:

$$\begin{pmatrix} \frac{-(m^2-1)}{(m^2+1)} & \frac{2m}{(m^2+1)} & \frac{-2mc}{(m^2+1)} \\ \frac{2m}{(m^2+1)} & \frac{(m^2-1)}{(m^2+1)} & \frac{2c}{(m^2+1)} \\ 0 & 0 & 1 \end{pmatrix}$$

where m= 1, c=1.

Q. Rotate triangle ABC by 450 clockwise about origin and scale it by (2,3) about origin.



Here,

The steps required are:

1. Rotate by 45 clockwise
2. scale by t_x= 2 and t_y= 3

Thus the composite matrix is given by

com = $S(2,3).R_{-45}$

$$= \begin{bmatrix} 2 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos 45 & Sin45 & 0 \\ -Sin45 & \cos 45 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 2 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ -1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 2/\sqrt{2} & 2/\sqrt{2} & 0 \\ -3/\sqrt{2} & 3/\sqrt{2} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Now, the required co-ordinate can be calculated as:

A' = com * A

$$= \begin{bmatrix} 2/\sqrt{2} & 2/\sqrt{2} & 0 \\ -3/\sqrt{2} & 3/\sqrt{2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 7 \\ 15 \\ 1 \end{bmatrix} =$$

B' = com * B

$$= \begin{bmatrix} 2/\sqrt{2} & 2/\sqrt{2} & 0 \\ -3/\sqrt{2} & 3/\sqrt{2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 5 \\ 8 \\ 1 \end{bmatrix} =$$

C' = com * C

$$= \begin{bmatrix} 2/\sqrt{2} & 2/\sqrt{2} & 0 \\ -3/\sqrt{2} & 3/\sqrt{2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 10 \\ 10 \\ 1 \end{bmatrix} =$$

Hence, the final co-ordinates are A'( ), B' ( ) and C'( ).

Q. Rotate the △ABC by 90° anti clockwise about (5,8) and scale it by (2,2) about (10,10)

Q. A mirror is placed such that it passes through (0,10) (10,0). Find the mirror image of an object (6,7), (7,6), (6,9).

Q. Show that Successive translations are additive

If two successive translations are applied then they are additive.

Proof:

If two successive translation vectors (t_{x1}, t_{y1}) and (t_{x2}, t_{y2}) are applied to a coordinate position P, the final transformed location P' is calculated with the following composite transformation as:

$$T' = T_{(x2,y2)} \cdot T_{(x1,y1)}$$

$$= \begin{bmatrix} 1 & 0 & Tx2 \\ 0 & 1 & Ty2 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & Tx1 \\ 0 & 1 & Ty1 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & Tx1+Tx2 \\ 0 & 1 & Ty1+Ty2 \\ 0 & 0 & 1 \end{bmatrix}$$

Hence, $T_{(tx2,\ tx2)}.T_{(tx1,ty1)} = T_{(tx1+ty1,\ tx2+ty2)}$ which demonstrates that two successive translation are additive

Q. Show that Successive rotation are additive

If two successive rotations are applied, then they are additive.

Proof:

Let P be point anticlockwise rotated by angle θ_1 to point P' and again let P' be rotated by angle θ_2 to point P'', then the combined transformation can be calculated with the following composite matrix as:

$$T = R_{(\theta 2)}\, R_{(\theta 1)}$$

$$= \begin{bmatrix} \cos\theta 2 & -\sin\theta 2 & 0 \\ \sin\theta 2 & \cos\theta 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta 1 & -\sin\theta 1 & 0 \\ \sin\theta 1 & \cos\theta 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \cos\theta 2 * cos\theta 1 - \sin\theta 2 * sin\theta 1 & -\cos\theta 2 * sin\theta 1 - sin\theta 2 * cos\theta 1 & 0 \\ \sin\theta 2 * cos\theta 1 + cos\theta 2 * sin\theta 1 & -\sin\theta 2 * sin\theta 1 + cos\theta 2 * cos\theta 2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \cos(\theta 1+\theta 2) & -\sin(\theta 1+\theta 2) & 0 \\ \sin(\theta 1+\theta 2) & \cos(\theta 1+\theta 2) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

i.e. $\mathbf{R}_{\theta 2}.\mathbf{R}_{\theta 1} = R(\theta_1 + \theta_2)$, which demonstrates that two successive rotations are additive.

Q. Show that Successive Scaling are multiplication

If two successive Scaling are applied, then they are multiplicative/commutative.

Proof:

Let point P is first scaled with scaling factor Sx1, Sx2 to Point P' and again let P' be scaled by scaling factor Sx2,Sy2 to Point P'', then the combined transformation can be calculated with the following composite matrix

$$T = S(S_{x2}, S_{y2})\, S(S_{x1}, S_{y1})$$

$$\begin{bmatrix} s_{x2} & 0 & 0 \\ 0 & s_{y2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_{x1} & 0 & 0 \\ 0 & s_{y1} & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} s_{x1}s_{x2} & 0 & 0 \\ 0 & s_{y1}s_{x2} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

i.e. $S(s_{x2,sy2})\, S(s_{x2,sy2}) = S(s_{x1.sx2},\ s_{y1.sy2})$, which demonstrates that two successive scaling are multiplicative.

Multiple Coordinate Systems in a Graphics Program

A coordinate system is a reference system used to represent the object along with its features within a common co-ordinate framework.

In a typical graphics program, we may need to deal with a number of different coordinate systems, and a good part of the work is the conversion of coordinates from one system to another. The list of some of the coordinate systems are:

a) **Modeling co-ordinate system**

The Model Coordinate System is simply the coordinate system where the model was created. It is used to define coordinates that are used to construct the shape of individual parts (objects) of a 2D scene

b) **World co-ordinate system**

A Model Coordinate System is the unique coordinate space of the model. Two distinct models, each with their own coordinate systems can't interact with each other. There needs to be a universal coordinate system that allows each model to interact with each other. This universal system is called *World Coordinate System*. For interaction to occur, the coordinate system of each model is transformed into the World Coordinate System

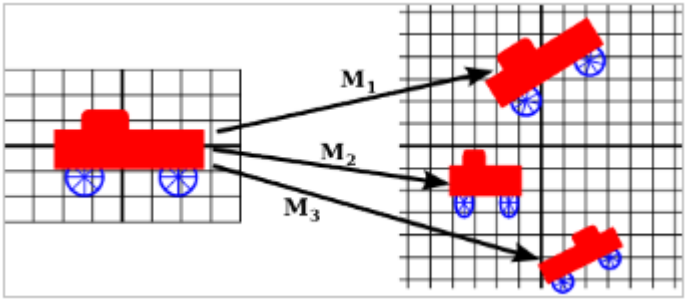


Fig: Modeling Coordinate Fig: World Coordinates

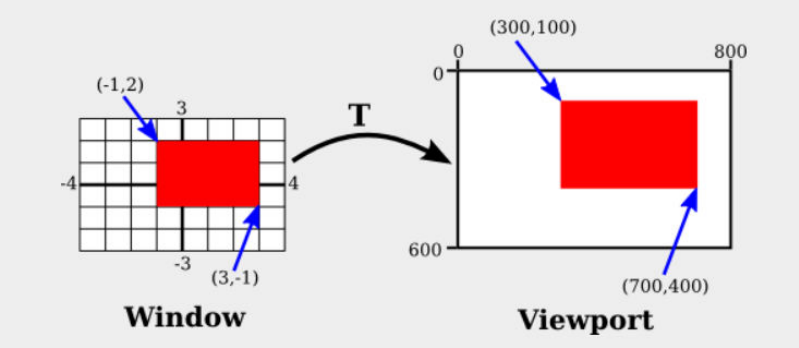
c) **Viewing co-ordinate system**

A world coordinate area selected for display is called **window**. Window is an area of picture that is selected for viewing. An area on display device to which a window is mapped is called **view**

port. View port is the part of computer screen.

Viewing co-ordinate system are used to define particular view of a 2D scene. Translation, scaling, and rotation of the window will generate a different view of the scene.

For a 2-D picture, a view is selected by specifying a sub area (window) of the total picture area.



d) **Normalized Viewing Co-ordinates**

NVC's are used to make the viewing process independent of the output device (monitor, mobile, hard copy devices). Normally, the value of NVC is 0 to 1.

e) **Device Co-ordinates**

Device co-ordinate are used to define coordinates in an output device. They are integers within the range (0, 0) to (x_{max}, y_{max}) for a particular output device.

Two Dimensional Viewing

The process of mapping the world co-ordinate scent to device co-ordinate is called viewing transformation or window to view port transformation or windowing transformation.

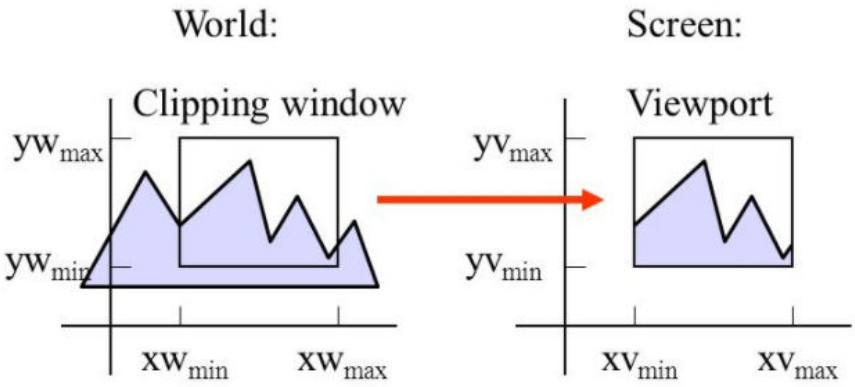


Fig: Mapping of picture section falling under rectangular Window onto a designated rectangular view port

The window defines what is to be viewed whereas the view port defines where it is to be displayed. Window can have any shape (circle, polygon) however some graphics package provide window and view port operations on standard rectangle only. Window deals with object space whereas view port deals with image space.

Transformations from world to device coordinates involve translation, rotation and scaling operations, as well as procedures for deleting those parts of the picture that are outside the limits of a selected display area i.e. clipping.

To make the viewing process independent of the requirements of any output device, graphics systems convert object descriptions to normalized coordinates.

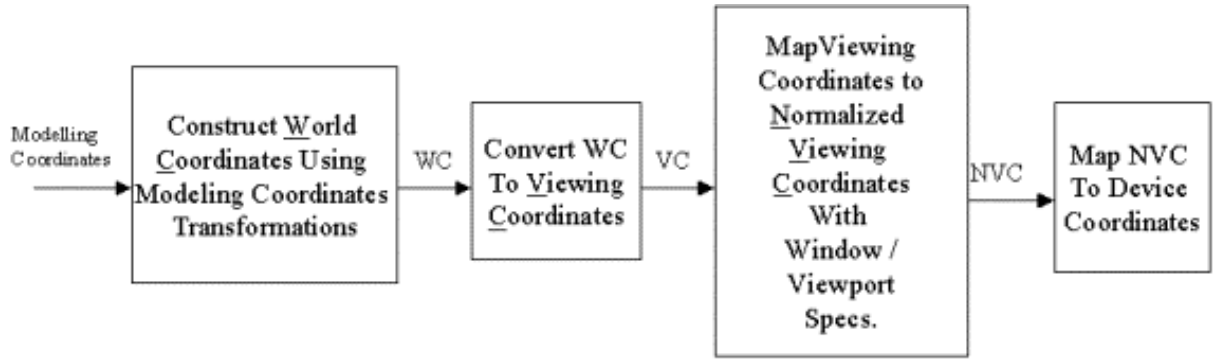


Fig: The 2D Viewing Transformation Pipeline

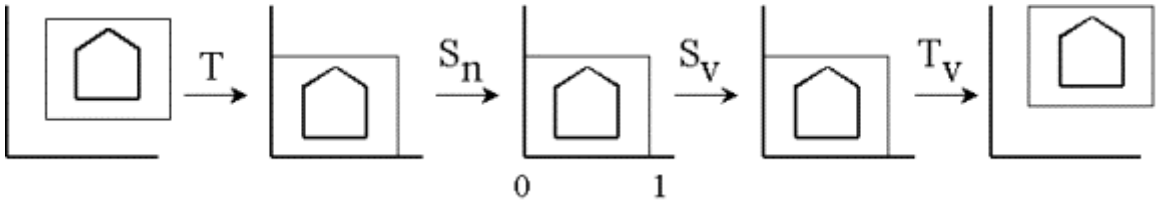
Procedure for to transform a window to the view port we have to perform the following steps:

Step1: The object together with its window is translated until the lower left corner of the window is at the origin

Step2: The object and window are scaled until the window has the dimensions of the view port

Step3: Again translate to move the view port to its correct position on the screen

(Setup Window,Translate window, Scale to normalize, Scale to view port, Translate to View port)

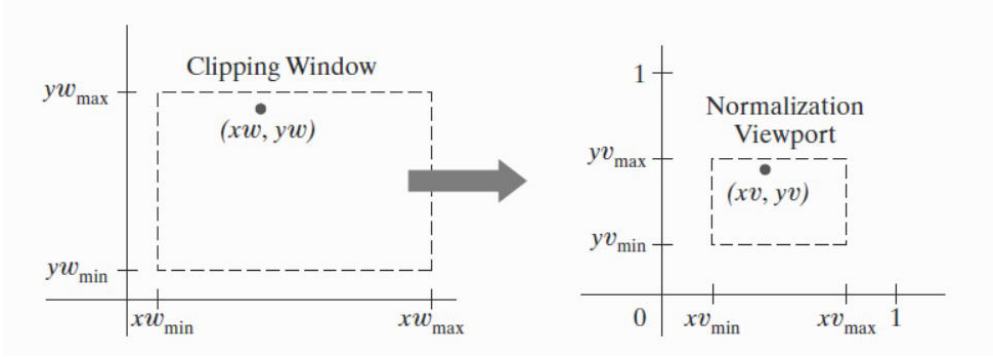


Application

i. By changing the position of the view port, we can view objects at different positions on the display area of an output device.
ii. By varying the size of view ports, we can change size of displayed objects.
iii. Zooming effects can be obtained by successively mapping different-sized windows on a fixed-sized view port
iv. Panning effects (Horizontal Scrolling) are produced by moving a fixed-size window across the various objects in a scene.

Window to View port Coordinate Transformation

A window can be specified by four world coordinates: xw_{min}, xw_{max}, yw_{min} and yw_{max}. Similarly, a view port can be described by four device coordinates: xv_{min}, xv_{max}, yv_{min} and yv_{max}



The following proportional ratios must be equal.

$$\frac{xv - xv_{min}}{xv_{max} - xv_{min}} = \frac{xw - xw_{min}}{xw_{max} - xw_{min}}$$

$$\frac{yv - yv_{min}}{yv_{max} - yv_{min}} = \frac{yw - yw_{min}}{yw_{max} - yw_{min}}$$

Solving these expressions, we get

$xv = xv_{min} + (xw - xw_{min})S_x$

$yv = yv_{min} + (yw - yw_{min})S_y$

where,

$$s_x = \frac{xv_{max} - xv_{min}}{xw_{max} - xw_{min}}$$

$$s_y = \frac{yv_{max} - yv_{min}}{yw_{max} - yw_{min}}$$

Q. Window port is given by (100,100,300,300) and view port is given by (50,50,150,150). Convert the window port coordinate (200,200) to the view port coordinate.

Solution:

$(xw_{min}, yw_{min}) = (100, 100)$

$(xw_{max}, xw_{max}) = (300, 300)$

$(xv_{min}, yv_{min}) = (50, 50)$

$(xv_{max}, xv_{max}) = (150, 150)$

(xw,yw)=(200,200)

Then, we have

$$s_x = \frac{xv_{max} - xv_{min}}{xw_{max} - xw_{min}} = (150 - 50)/ (300 - 100) = 0.5$$

$$s_y = \frac{yv_{max} - yv_{min}}{yw_{max} - yw_{min}} = (150 - 50)/ (300 - 100) = 0.5$$

The equations for mapping window co-ordinate to view port coordinate is given by

$xv = xv_{min} + (xw - xw_{min})S_x$

$yv = yv_{min} + (yw - yw_{min})S_y$

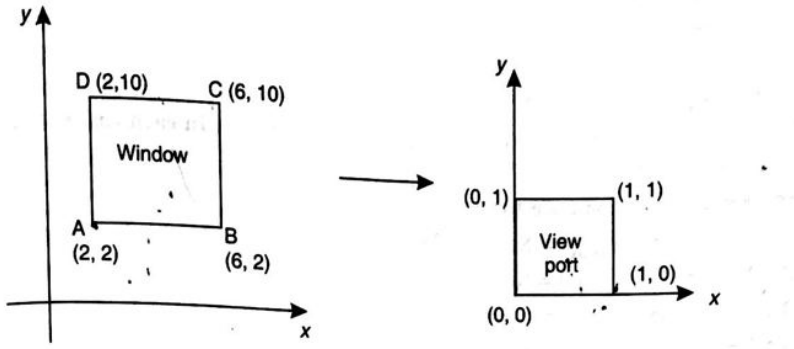
Hence,

$xv = 50 + (200 - 100) * 0.5 = 100$

$yv = 50 + (200 - 100) * 0.5 = 100$

The transformed view port coordinate is (100,100).

Q. Find the normalization transformation matrix for window to view port which uses the rectangle whose lower left corner is at (2,2) and upper right corner is at (6,10) as a window and the view port that has lower left corner at (0,0) and upper right corner at (1,1)



The composite transformation matrix for transforming the window co-ordinate to viewport coordinate is given as

$$T = S(s_x, s_y)T(-2, -2)$$

Now we know,

$$s_x = \frac{xv_{max} - xv_{min}}{xw_{max} - xw_{min}}$$

$$s_y = \frac{yv_{max} - yv_{min}}{yw_{max} - yw_{min}}$$

$S_x = (1\text{-}0)/(6 - 2) = 0.25$
$S_y = (1\text{-}0)/(10 - 2) = 0.125$

Then, transformation matrix,

$$T = \begin{bmatrix} Sx & 0 & 0 \\ 0 & Sy & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & tx \\ 0 & 1 & ty \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0.25 & 0 & 0 \\ 0 & 0.125 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -2 \\ 0 & 1 & -2 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0.25 & 0 & -0.5 \\ 0 & 0.125 & -0.25 \\ 0 & 0 & 1 \end{bmatrix}$$

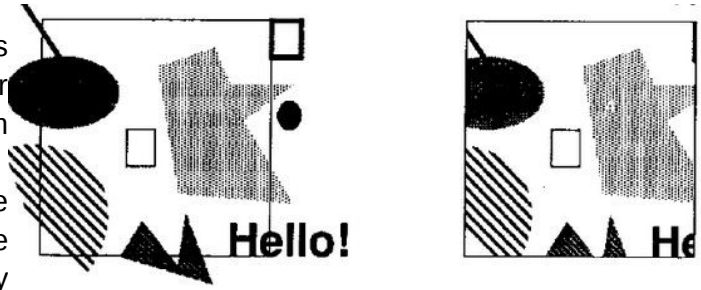
Clipping

The process of discarding those parts of a picture which are outside of a specified region or window is called **clipping**. The procedure using which we can identify whether the portions of the graphics object is within or outside a specified region or space is called **clipping algorithm**.

The region or space which is used to see the object is called window and the region on which the object is shown is called **view port**.

Clipping is necessary to remove those portions of the objects which are not necessary for further operation's. excludes unwanted graphics from the screen. So there are three cases

i. The object may be completely outside the viewing area defined by



the window port.

ii. The object may be seen partially in the window port.

iii. The object may be seen completely in the window port

For case i and ii, clipping operation is necessary but not for case iii.

Applications of Clipping

i. Extracting part of a defined scene for viewing.

ii. Identifying visible surfaces in three-dimensional views.

iii. Anti aliasing line segments or object boundaries.

iv. Drawing and painting operations that allow parts of a picture to be selected for copying, moving erasing, or duplicating.

Types of Clipping

i. Point Clipping

ii. Line Clipping (straight-line segments)

iii. Area Clipping (polygons)

iv. Curve Clipping

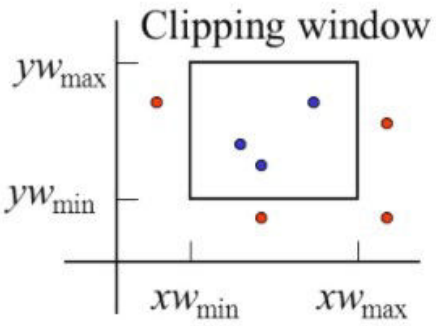
v. Text Clipping

Point Clipping

Point clipping is the process of removing all those points that lies outside a given region or window. Let xw_{min} and xw_{max} be the edge of the boundaries of the clipping window parallel to Y axis and yw_{min} and yw_{max} be the edge of the boundaries of the clipping window parallel to X axis as shown in figure below.
So any point P(x,y) of world coordinate can be saved for display if the following conditions are satisfied

$$xw_{\min} \le x \le xw_{\max}$$

$$yw_{\min} \le y \le yw_{\max}$$



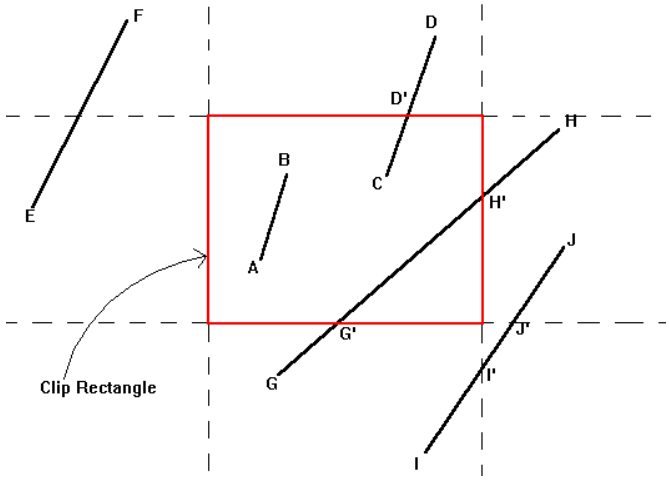
Line Clipping

In line clipping, a line or part of line is clipped if it is outside the window port. All the line segment falls into one of the following clipping cases.

a. The line is totally outside the window port so is directly clipped.

b. The line is partially clipped if a part of the line lies outside the window port.

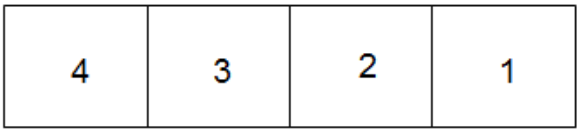
c. The line is not clipped if all whole line lied within the window port.

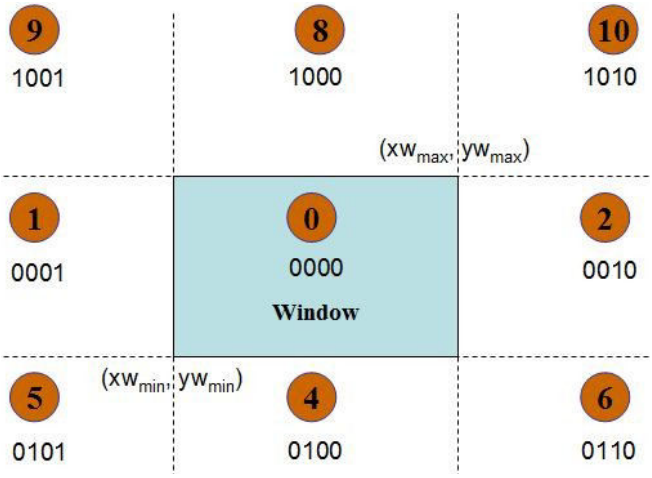


So in the above figure, line FE require total clipping, CD, GH, IJ require partial clipping and AB requires no clipping.

Cohen-Sutherland Line Clipping Algorithm

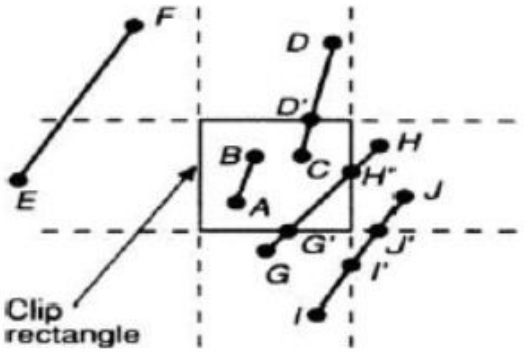
- Divide 2D space into 3x3 = 9-regions.
- Middle region is the clipping window.
- Each region is assigned a 4-bit code.
- Bit 1 is set to 1 if the region is to the left of the clipping window, otherwise.
  Similarly we deal for bits 2, 3 and 4.





- Any point inside the clipping window has a region code 0000.
- Any endpoint (x, y) of a line segment, the code can be determined as follows:
  - If $x < xw_{min}$, first bit is 1, (Point lies to left of window(Left)) (0th bit) Otherwise 0.
  - If $x > xw_{max}$, second bit is 1, (Point lies to right of window(Right)) (1st bit), otherwise 0.
  - If $y < yw_{min}$, third bit is 1, (Point lies to below window(Bottom)) (2nd bit), otherwise 0.
  - If $y > yw_{max}$, fourth bit is 1, (Point lies to above window(Top)) (3rd bit), otherwise 0.

Example:



Line	Code 1	Code 2
A⇒B	0000	0000
C⇒D	0000	1000
E⇒F	0001	1001
G⇒H	0100	0010
I⇒J	0100	0010

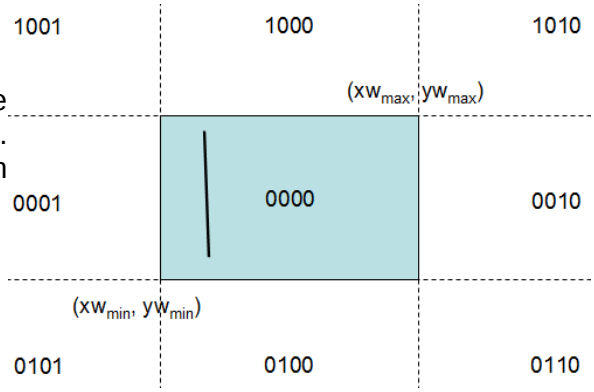
Algorithm:

a) Given a line segment with end points P1=(x1,y1) and P2(x2,y2), compute 4 bit region code for each end point.

b) To clip a line, first we have to find out which regions its two endpoints lie in.

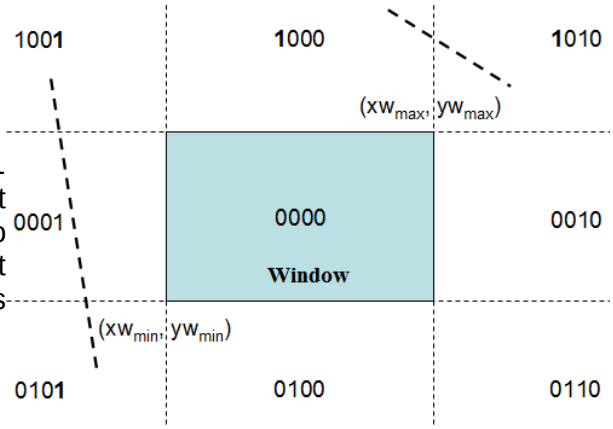
Case I:

If both end point code is 0000, then the line segment is completely inside the window. i.e. if bitwise OR of the codes yields 0000, then the line segment is accepted for display.



Case II:

If the two end point region code both have a 1 in the same bit position, then the line segment lies completely outside the window, so discarded. i.e. if bitwise AND of the codes not equal to 0000, then the line segment is rejected.



Case III:

If lines can not be identified as completely inside or outside we have to do some more calculations.

Here we find the intersection points with a clipping boundary using the slope intercept form of the line equation

Here, to find the visible surface, the intersection points on the boundary of window can be determined as:

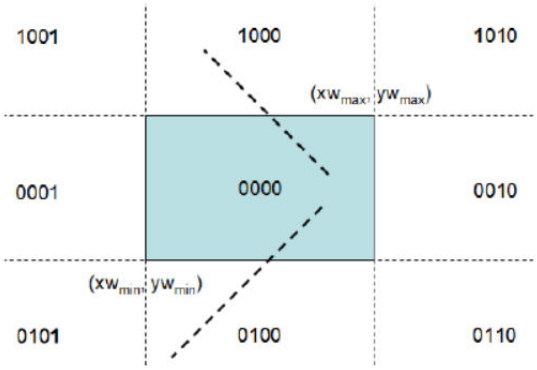
$$y - y_1 = m\,(x - x_1)$$

where $m = (y_2 - y_1)/(x_2 - x_1)$

i. If the intersection is with vertical boundary

$$x = x_1 + (y - y_1)/m$$

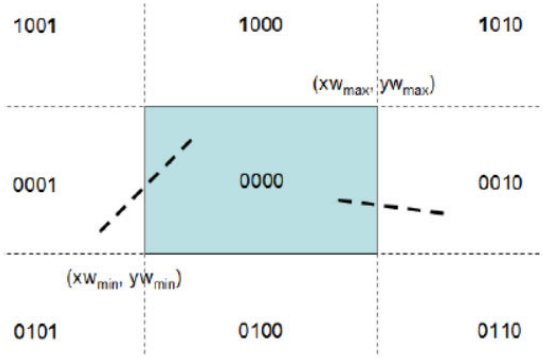
Where y is set to either yw_{min} or yw_{max}



ii. If the intersection is with vertical boundary

$$y = y_1 + m\,(x - x_1)$$

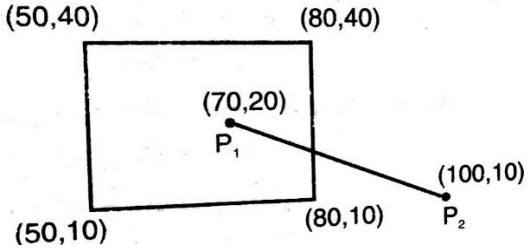
Where x is set to either Xw_{min} or Xw_{max}



c) Assign a new four-bit code to the intersection point and repeat until either cas1 or case2 are satisfied.

Q. Use the Cohen –Sutherland algorithm to clip the line P1(70,20) and P2(100,10) against a window lower left hand corner (50,10) and upper right hand corner (80,40)

Solution:



Assigning 4 bit binary code to the two end point

P1 = 0000

P2 = 0010

Finding bitwise OR:

P1 | P2 = 0000 | 0010 = 0010

Since P1 | P2 ! = 0000,hence the two point doesn't lie completely inside the window.

Finding bitwise AND:

P1 & P2 = 0000 & 0010 = 0000

since P1 & P2 = 0000, hence line is partially visible.

Now, For finding the intersection of P1 and P2 with the boundary of Window,
We have,
P1(x1,y1) = (70,20)
P2(x2,y2) = (100,10)
Slope m = (10-20)/(100-70) = -1/3
We have to find the intersection with right edge of window, here x=80, y=?
We have

y = y2 + m(x-x2)
= 10 +(-1/3)(80-100)
= 10 + 6.67
= 16.67

Thus the intersection point P3 = (80,16.66), So discarding the line segment that lie outside the boundary i.e P3P2, we get new line P1P3 with co-ordinate P1(70, 20) and P3(80,16.67)

Polygon Clipping: Sutherland – Hodgeman

The Sutherland–Hodgeman algorithm is used for clipping polygons. A single polygon can actually be split into multiple polygons. The algorithm clips a polygon against all edges of the clipping region in turn. This algorithm is actually quite general — the clip region can be any convex polygon in 2D, or any convex polyhedron in 3D.

There are four possible cases for any given edge of given polygon against clipping edge.

1. **Both vertices are inside** :
   Only the second vertex is added to the output list
2. **First vertex is outside while second one is inside** :
   Both the point of intersection of the edge with the clip boundary and the second vertex are added to the output list
3. **First vertex is inside while second one is outside** :
   Only the point of intersection of the edge with the clip boundary is added to the output list
4. **Both vertices are outside** :

   No vertices are added to the output list

Case	1st vertex	2nd vertex	output
1	inside	inside	2nd vertex
2	inside	outside	intersection
3	outside	outside	none
4	outside	inside	2nd and intersection

Example:

From	To	1st point	2nd point	Case	Output list
V_1	V_2	Outside	Inside	4	V'_1 and V_2
V_2	V_3	Inside	Outside	2	V'_2
V_3	V_1	Outside	Outside	3