FUTURE VISION BIE

One Stop for All Study Materials
& Lab Programs

By K B Hemanth Raj

Scan the QR Code to Visit the Web Page

Or

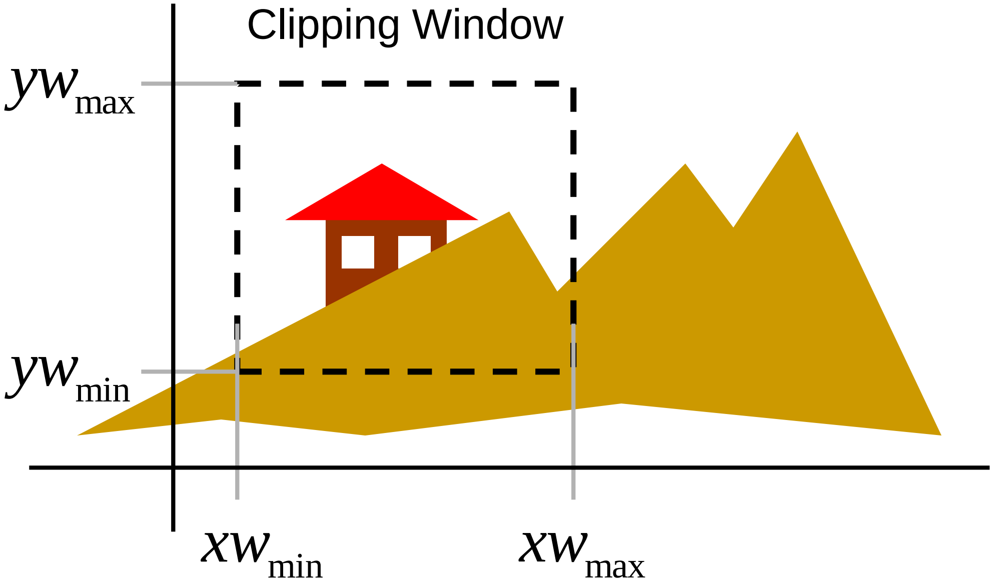
Visit : https://hemanthrajhemu.github.io

Gain Access to All Study Materials according to VTU,
CSE – Computer Science Engineering,
ISE – Information Science Engineering,
ECE - Electronics and Communication Engineering
& MORE...

Window to Viewport Transformation

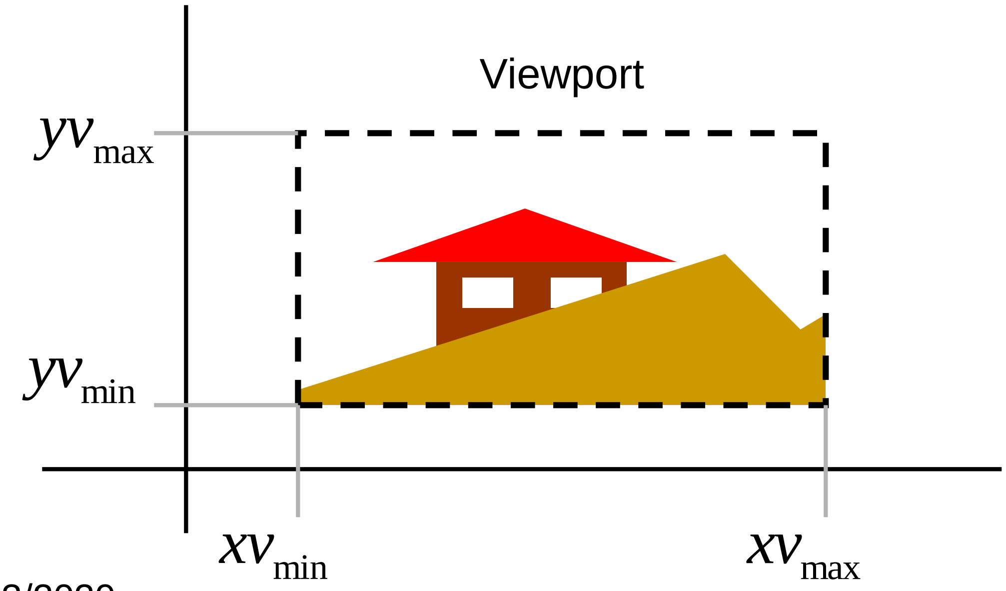
Viewing

- Transformation world$\rightarrow$screen
- Clipping: Removing parts outside screen



World Coordinates

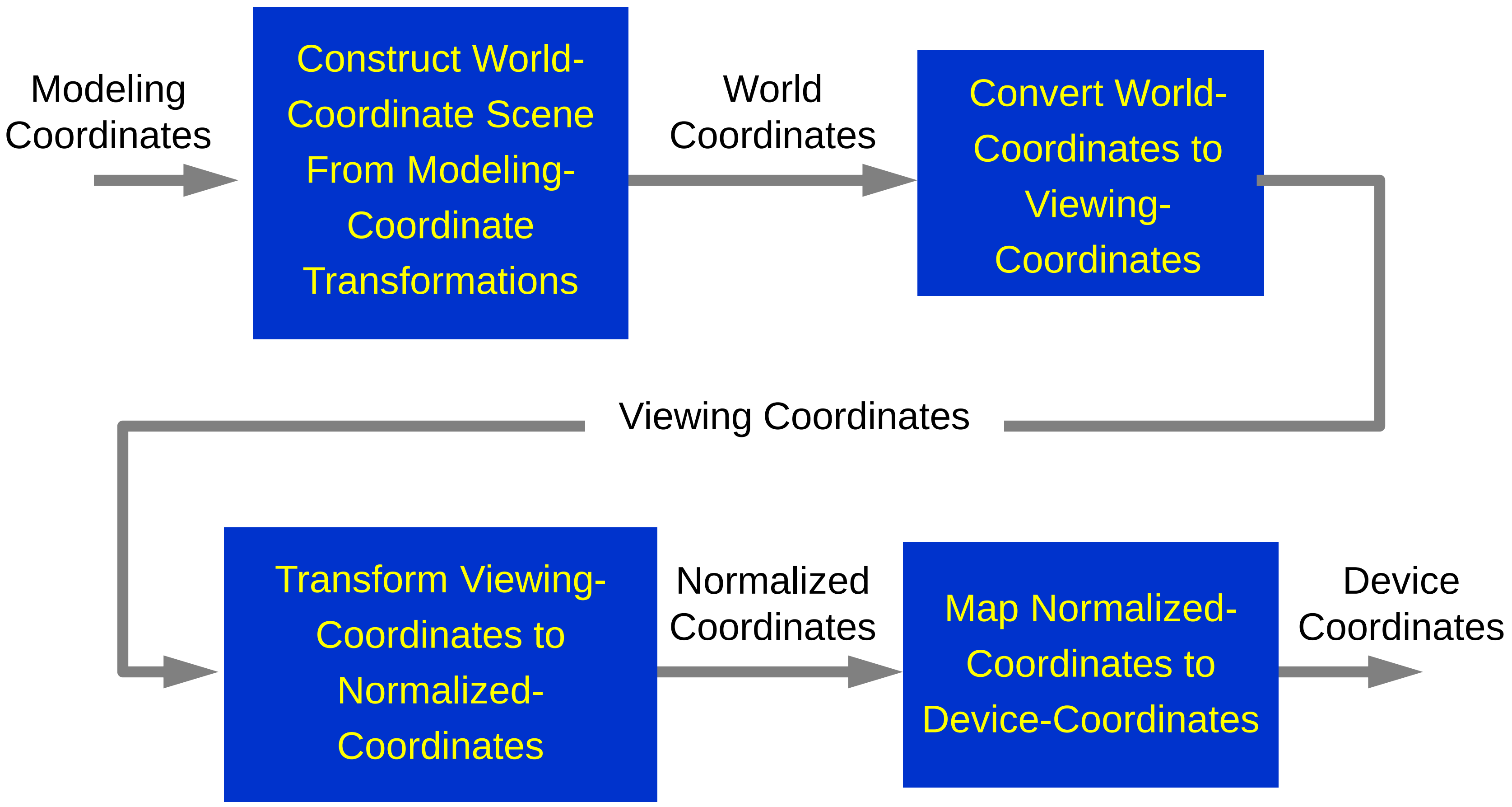
The clipping window is mapped into a viewport.



Viewing world has its own coordinates, which may be a non-uniform scaling of world coordinates.

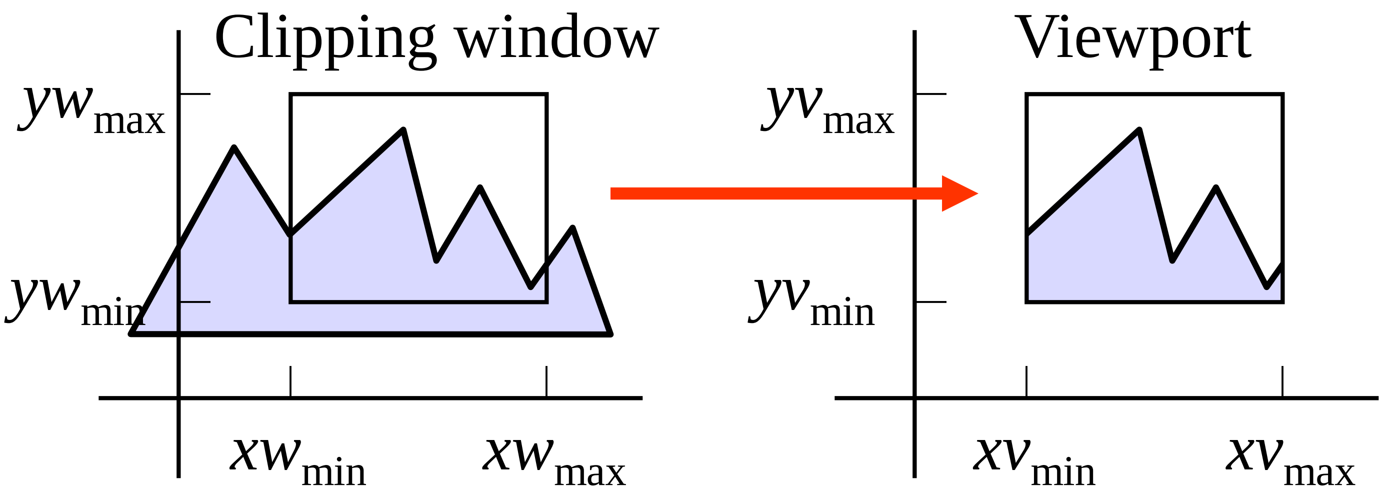
Viewport Coordinates

2D viewing transformation pipeline



2D Viewing pipeline

World: Screen:



Clipping window:

What do we want to see?

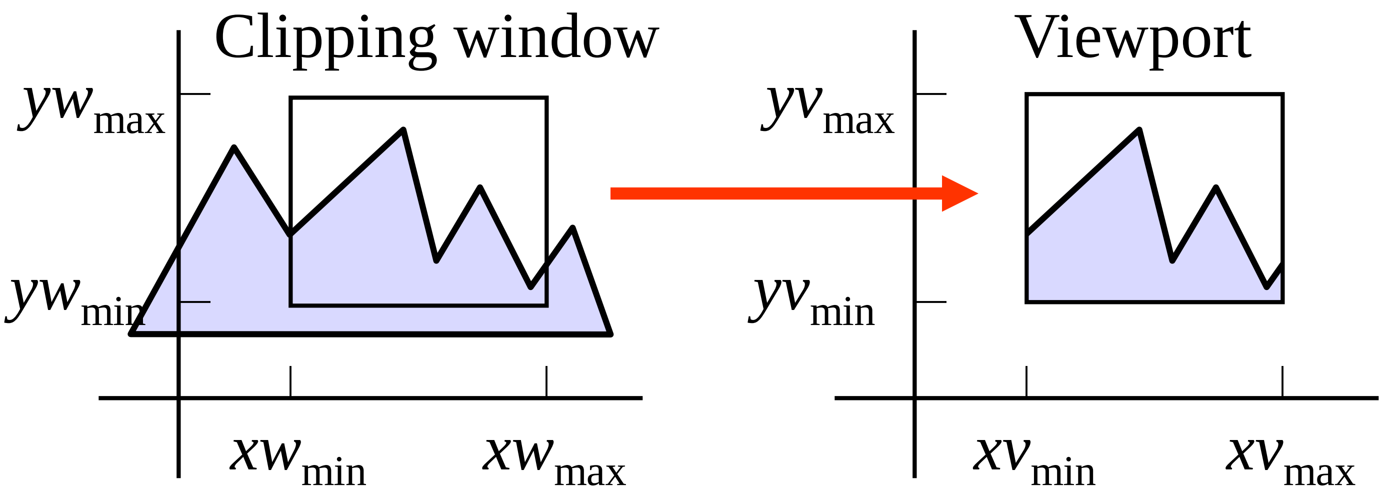
Viewport:

Where do we want to see it?

2D Viewing pipeline

World:

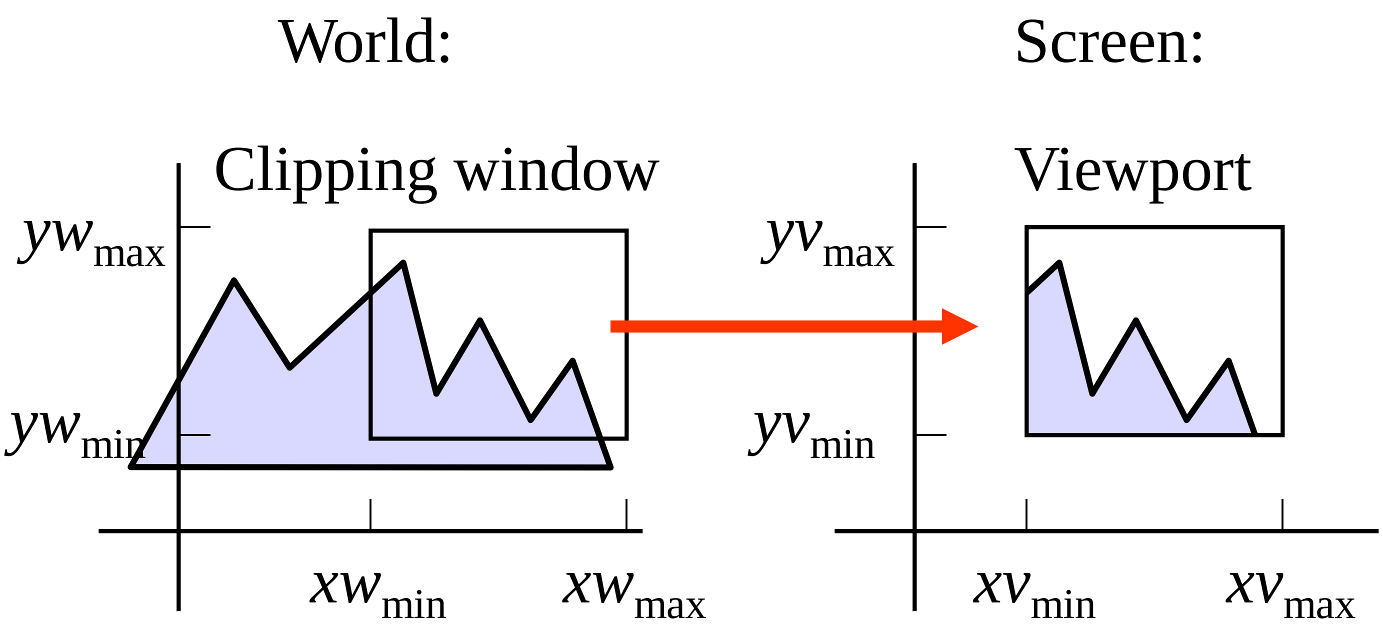
Screen:



Clipping window:

Panning…

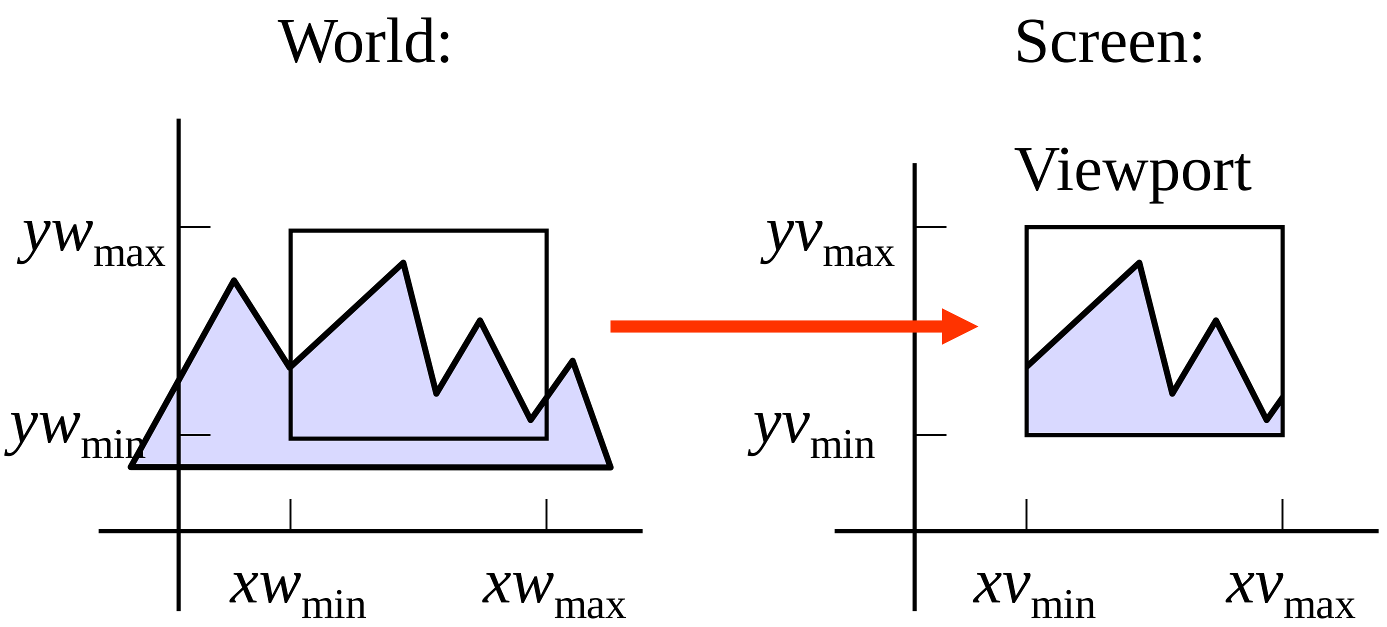
2D Viewing pipeline



Clipping window:

Panning…

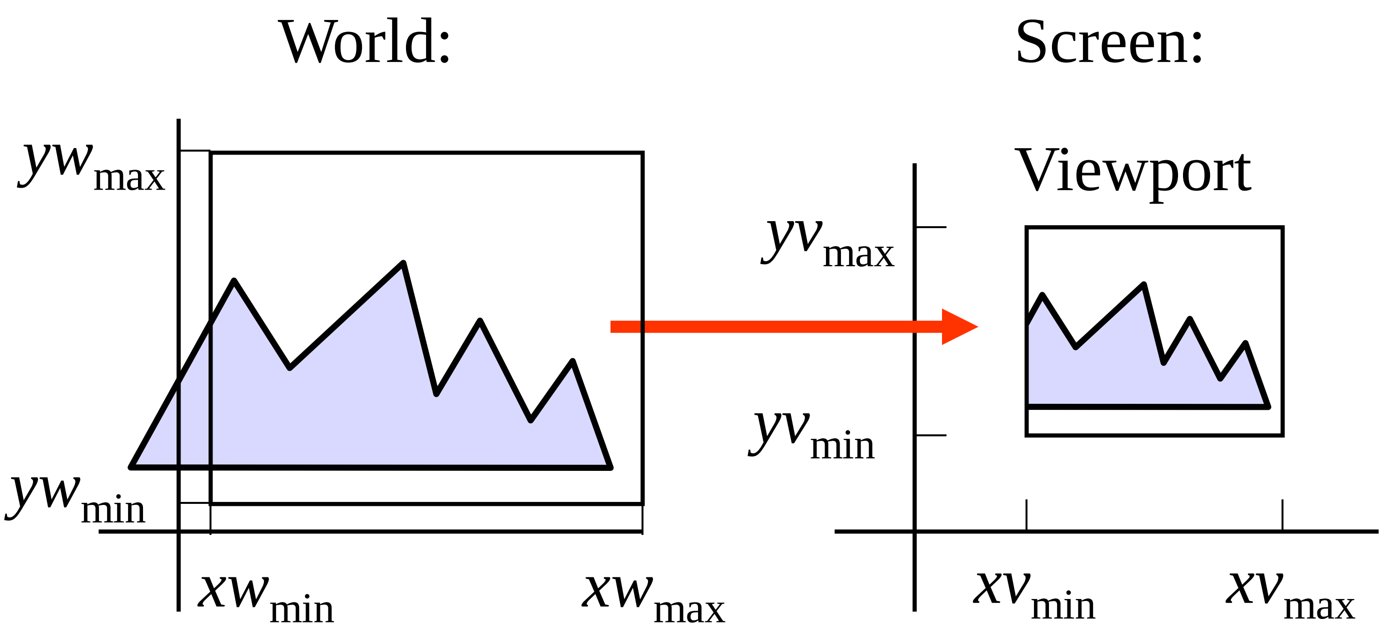
2D Viewing pipeline



Clipping window:

Zooming…

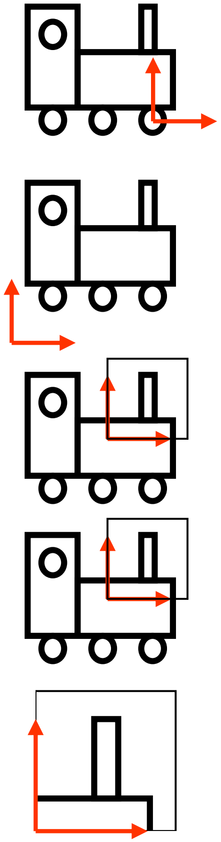
2D Viewing pipeline



Clipping window:

Zooming…

2D Viewing pipeline



MC: Modeling Coordinates

↓ *Apply model transformations*

WC: World Coordinates

↓ *Determine visible parts*

VC: Viewing Coordinates

↓ *To standard coordinates*

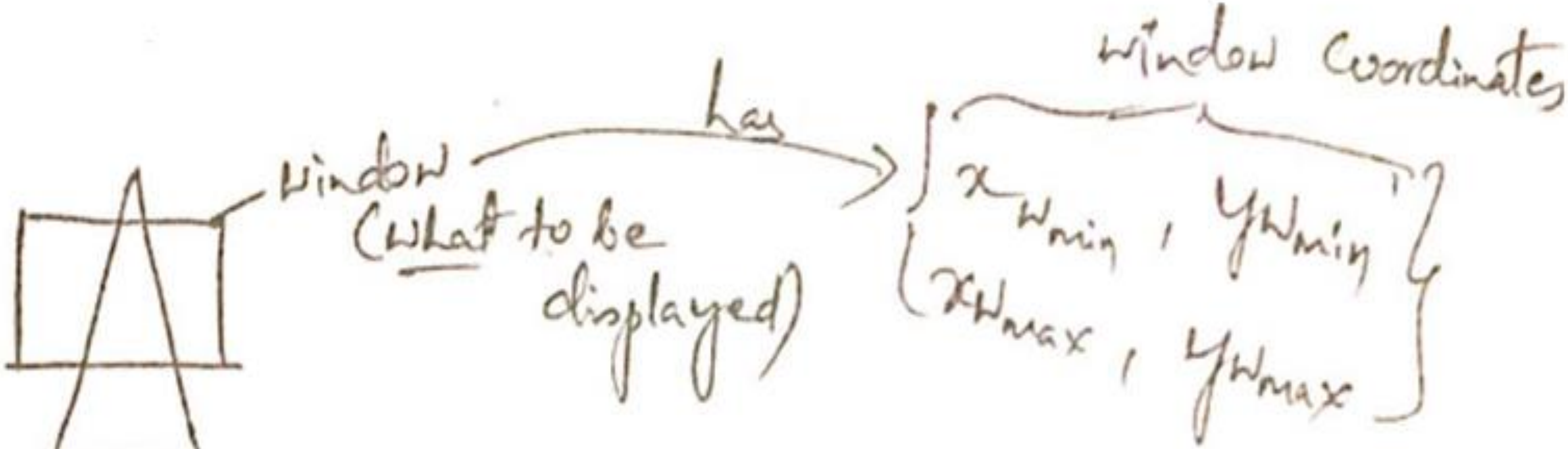
NC: Normalized Coordinates

↓ *Clip and determine pixels*

DC: Device Coordinates

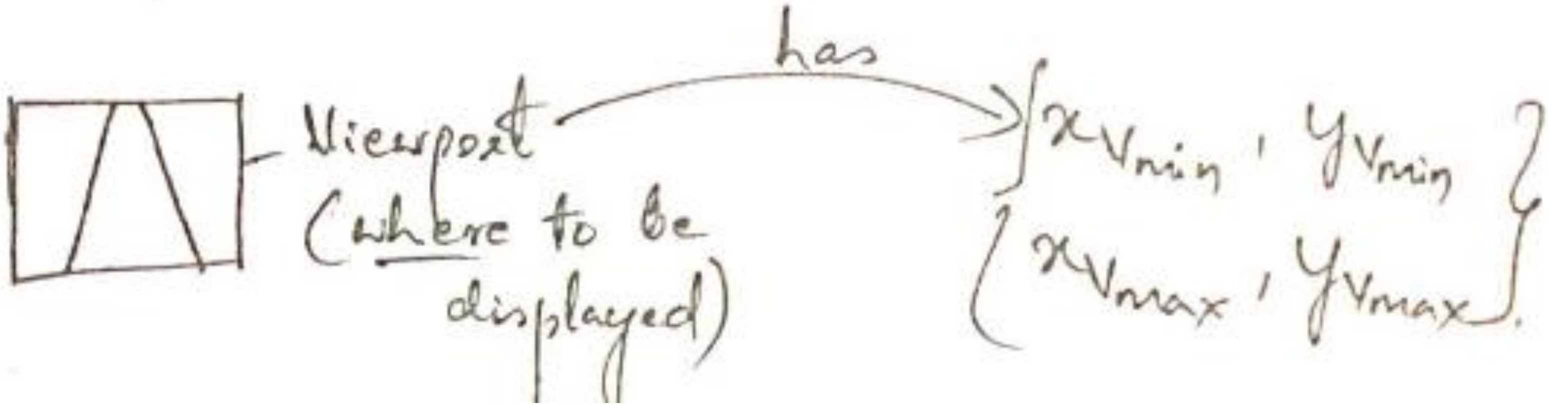
Window to Viewport Transformation

- What is window?



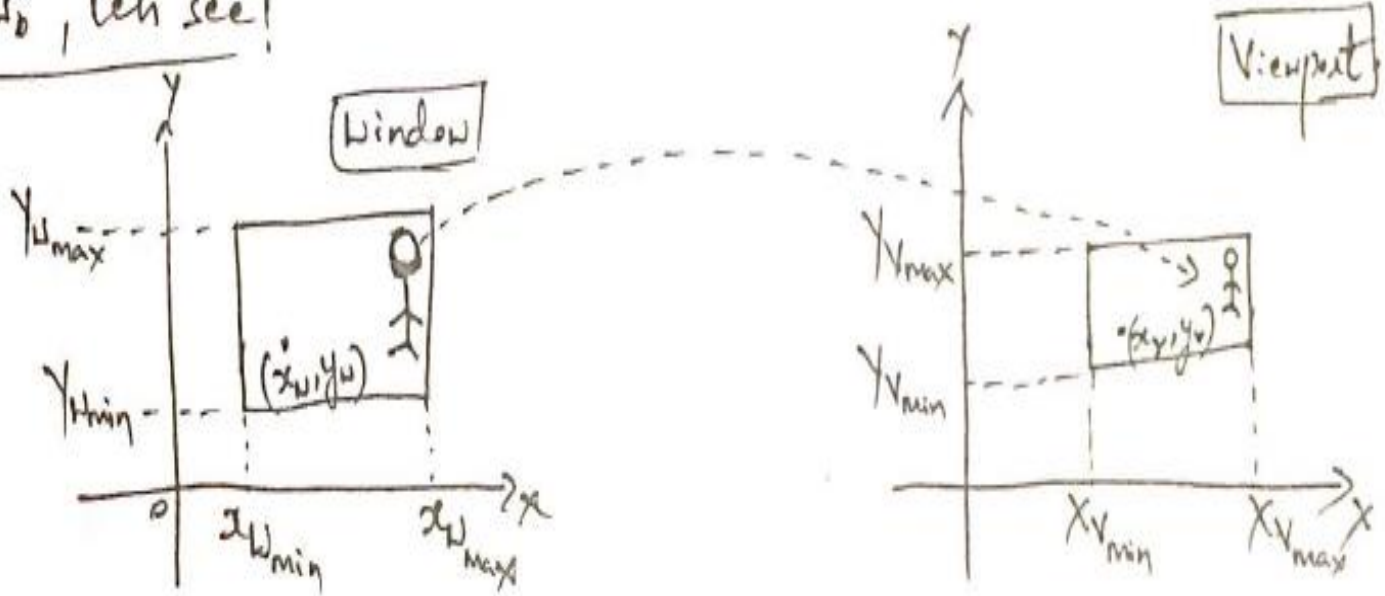
Window to Viewport Transformation

- What is viewport?



So, we have to map (convert) window to Viewport coordinat

How?, let's see!



Relative position will be same for both window & Viewport, but the size of the object changes.

Since the relative position is same, we can have

for x →
$$\frac{x_w - x_{w_{min}}}{x_{w_{max}} - x_{w_{min}}} = \frac{x_v - x_{v_{min}}}{x_{v_{max}} - x_{v_{min}}} \quad — (1)$$

for y →
$$\frac{y_w - y_{w_{min}}}{y_{w_{max}} - y_{w_{min}}} = \frac{y_v - y_{v_{min}}}{y_{v_{max}} - y_{v_{min}}} \quad — (2)$$

But, what we have to find!?
the corresponding viewport coordinates. So
find out x_v, y_v from above equations.

So, from (i) $\Rightarrow x_v - x_{v_{min}} = (x_{v_{max}} - x_{v_{min}}) \left(\frac{x_w - x_{w_{min}}}{x_{w_{max}} - x_{w_{min}}} \right)$

$$\Rightarrow x_v - x_{v_{min}} = (x_w - x_{w_{min}}) \left(\frac{x_{v_{max}} - x_{v_{min}}}{x_{w_{max}} - x_{w_{min}}} \right)$$

leave some 4 lines gap here

$$\Rightarrow x_v - x_{v_{min}} = x_w\left(\frac{x_{v_{max}} - x_{v_{min}}}{x_{w_{max}} - x_{w_{min}}}\right) - x_{w_{min}}\left(\frac{x_{v_{max}} - x_{v_{min}}}{x_{w_{max}} - x_{w_{min}}}\right)$$

$$\Rightarrow x_v - x_{v_{min}} = x_w\left(\frac{x_{v_{max}} - x_{v_{min}}}{x_{w_{max}} - x_{w_{min}}}\right) - \frac{x_{w_{min}} x_{v_{max}} + x_{w_{min}} x_{v_{min}}}{x_{w_{max}} - x_{w_{min}}}$$

$$\Rightarrow x_v = x_w\left(\frac{x_{v_{max}} - x_{v_{min}}}{x_{w_{max}} - x_{w_{min}}}\right) + x_{v_{min}} + \frac{x_{w_{min}} x_{v_{min}} - x_{w_{min}} x_{v_{max}}}{x_{w_{max}} - x_{w_{min}}}$$

$$2)\; x_v = x_w\left(\frac{x_{v_{max}} - x_{v_{min}}}{x_{w_{max}} - x_{w_{min}}}\right) + \frac{x_{v_{min}}\left(x_{w_{max}} - x_{w_{min}}\right) + x_{w_{min}} x_{v_{min}} - x_{w_{min}} x_{v_{max}}}{x_{w_{max}} - x_{w_{min}}}$$

$$\Rightarrow\; x_v = x_w\left(\frac{x_{v_{max}} - x_{v_{min}}}{x_{w_{max}} - x_{w_{min}}}\right) + \frac{x_{v_{min}} x_{w_{max}} - \cancel{x_{v_{min}} x_{w_{min}}} + \cancel{x_{w_{min}} x_{v_{min}}} - x_{w_{min}} x_{v_{max}}}{x_{w_{max}} - x_{w_{min}}}$$

$$\Rightarrow x_v = x_w \underbrace{\left(\frac{x_{v_{max}} - x_{v_{min}}}{x_{w_{max}} - x_{w_{min}}}\right)}_{} + \underbrace{\left(\frac{x_{w_{max}} x_{v_{min}} - x_{w_{min}} x_{v_{max}}}{x_{w_{max}} - x_{w_{min}}}\right)}_{}$$

$$\Rightarrow \boxed{x_v = x_w S_x + T_x}$$

where $\boxed{S_x = \frac{x_{v_{max}} - x_{v_{min}}}{x_{w_{max}} - x_{w_{min}}}}$

$$\boxed{T_x = \frac{x_{w_{max}} x_{v_{min}} - x_{w_{min}} x_{v_{max}}}{x_{w_{max}} - x_{w_{min}}}}$$

final

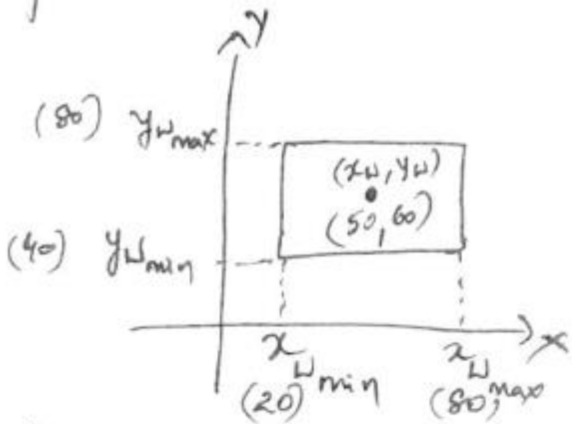
Similarly, do for Yv

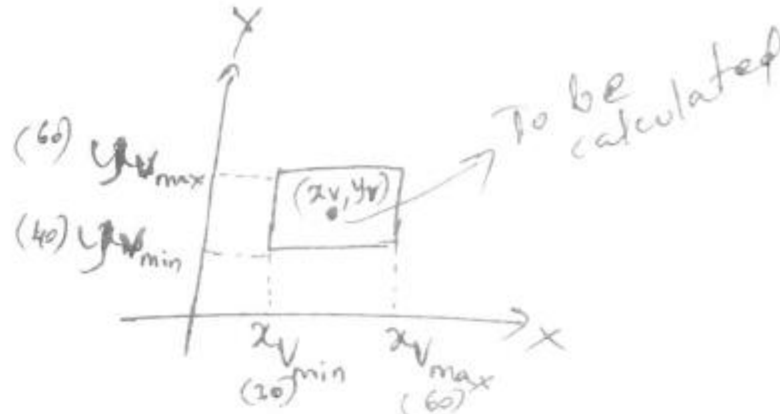
$$\boxed{y_v = y_w S_y + T_y}$$

where
$$\boxed{S_y = \frac{y_{v_{max}} - x_{v_{min}}}{y_{w_{max}} - y_{w_{min}}}}$$

$$\boxed{T_y = \frac{y_{w_{max}}\, y_{v_{min}} - y_{w_{min}}\, y_{v_{max}}}{y_{w_{max}} - y_{w_{min}}}}$$

Example Problem





Given:-

$x_{w_{min}} = 20$

$x_{w_{max}} = 80$

$y_{w_{min}} = 40$

$y_{w_{max}} = 80$

$(x_w, y_w) = (50, 60)$

$x_{v_{min}} = 30$

$x_{v_{max}} = 60$

$y_{v_{min}} = 40$

$y_{v_{max}} = 60$

$(x_v, y_v) = ?$

To find out? (x_v, y_v)

substitute the given in 1 & 2

$$1 \Rightarrow \frac{x_v - 30}{60 - 30} = \frac{50 - 20}{80 - 20}$$

$$x_v - 30 = 15$$

$$\boxed{x_v = 45}$$

② $\Rightarrow$ $$\frac{y_v - 40}{60 - 40} = \frac{60 - 40}{80 - 40}$$

$$y_v - 40 = 10$$

$$\boxed{y_v = 50}$$

Conclusion:-

An object which was at $(50, 60)$ in world coordinates [x_w, y_w], when captured by the camera, it got placed at the screen coordinates (x_v, y_v) at $(45, 50)$.

Now, go back to that gap...

$$x_v - x_{v_{min}} = \left(x_w - x_{w_{min}}\right)\left(\frac{x_{v_{max}} - x_{v_{min}}}{x_{w_{max}} - x_{w_{min}}}\right)$$

$$\therefore \boxed{x_v = x_{v_{min}} + \left(x_w - x_{w_{min}}\right) S_x}$$

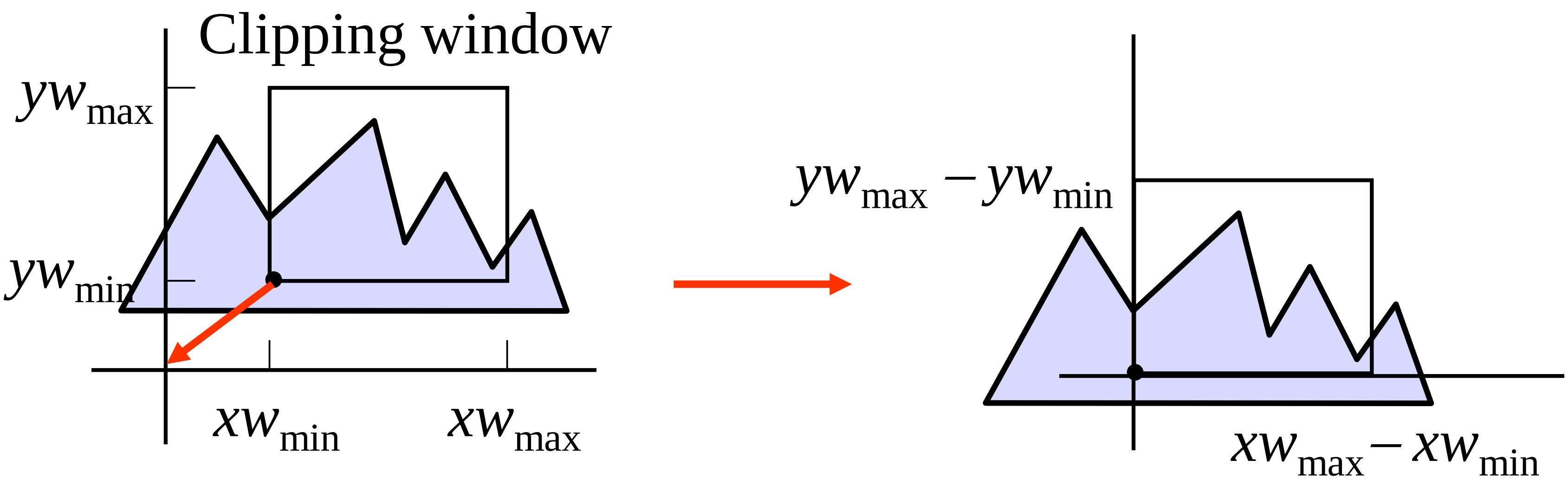
similarly,

$$\boxed{y_v = y_{v_{min}} + \left(y_w - y_{w_{min}}\right) S_y}$$

Apparently, in 5th lab program (cohen Sutherland line clip)

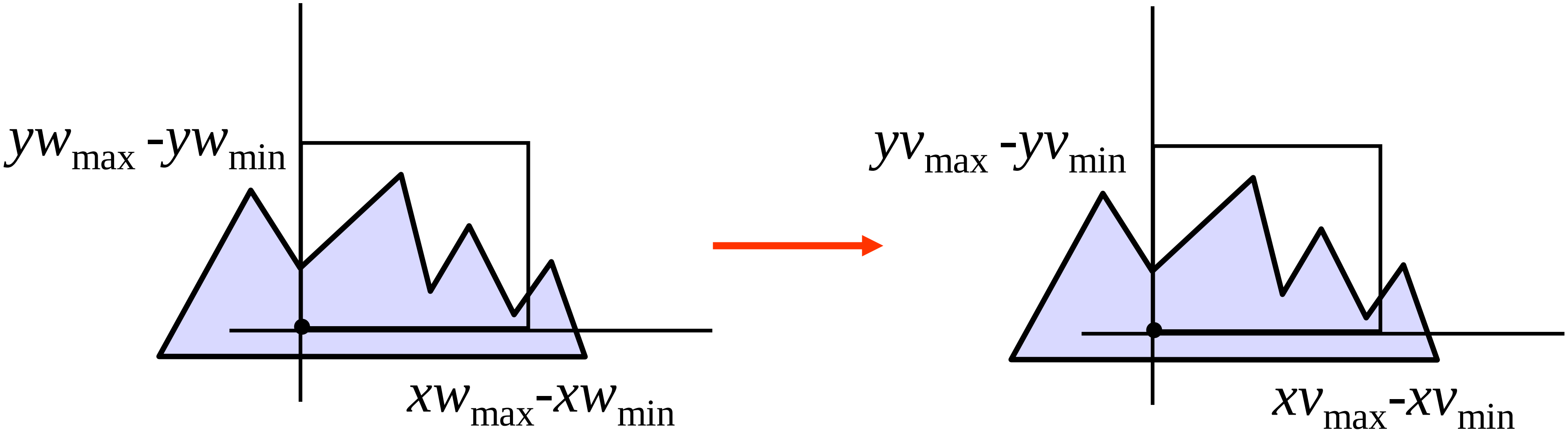
```
double sx=(xvmax-xvmin)/(xmax-xmin);
double sy=(yvmax-yvmin)/(ymax-ymin);
double vx0=xvmin+(x0-xmin)*sx;
double vy0=yvmin+(y0-ymin)*sy;
double vx1=xvmin+(x1-xmin)*sx;
double vy1=yvmin+(y1-ymin)*sy;
```

Clipping window



- Clipping window usually an *axis-aligned* rectangle
- Sometimes rotation
- From world to view coordinates: $\mathbf{T}\left(-xw_{\min}, -yw_{\min}\right)$ possibly followed by rotation
- More complex in 3D

To normalized coordinates



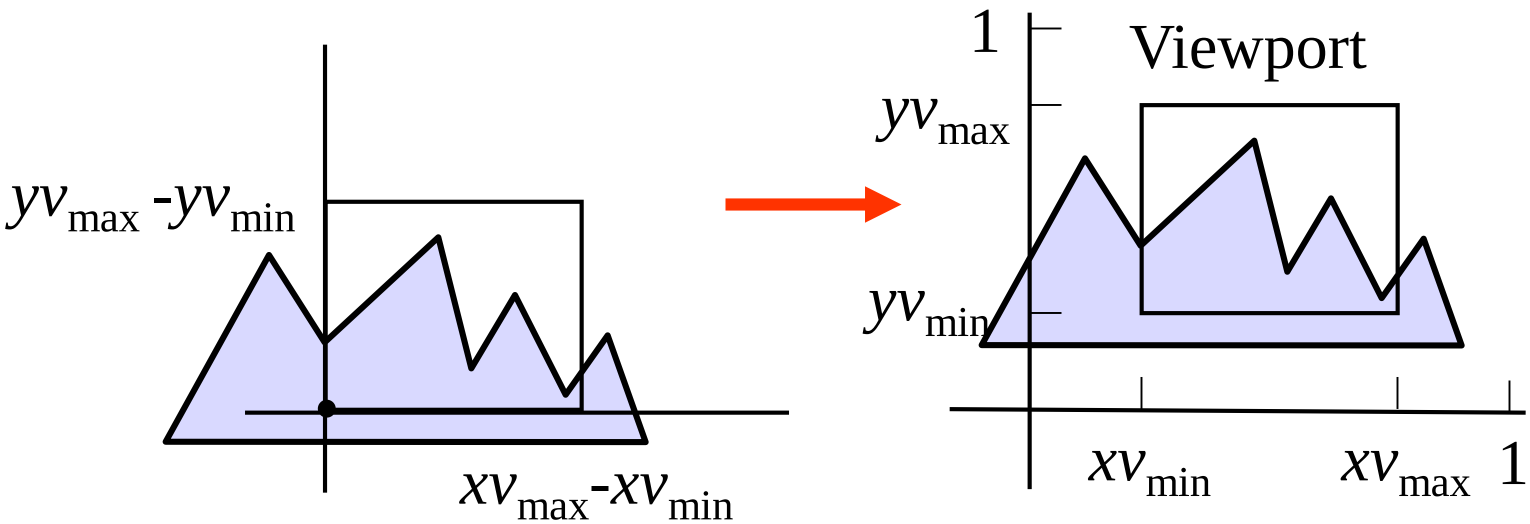
Scale with :

$$\mathbf{S}\left(\frac{xv_{max}-xv_{min}}{xw_{max}-xw_{min}},\frac{yv_{max}-yv_{min}}{yw_{max}-yw_{min}}\right)$$

If the two scale factors are unequal,

then the aspect-ratio changes: distortion!

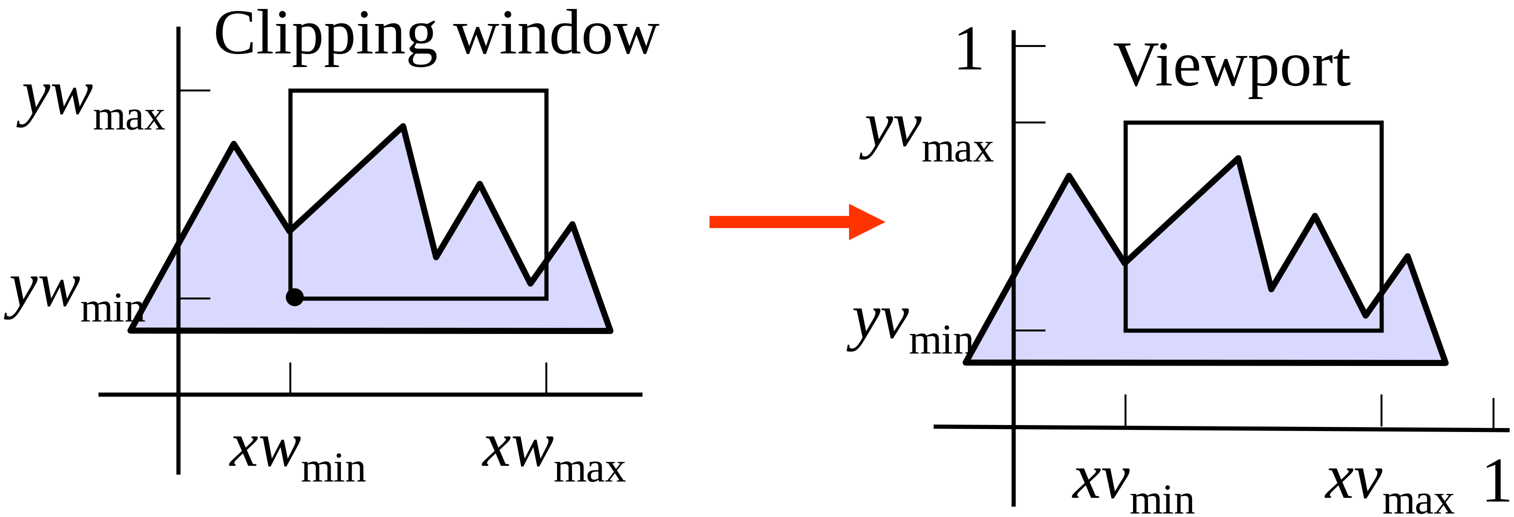
To normalized coordinates



Translate with:

$\mathbf{T}\left(xv_{min}, yv_{min}\right)$

To normalized coordinates



All together:

$$\mathbf{T}(xv_{min}, yv_{min})\mathbf{S}\left(\frac{xv_{max} - xv_{min}}{xw_{max} - xw_{min}}, \frac{yv_{max} - yv_{min}}{yw_{max} - yw_{min}}\right)\mathbf{T}(-xw_{min}, -yw_{min})$$

OpenGL 2D Viewing

Specification of 2D Viewing in OpenGL:

- Standard pattern, follows terminology.

First, this is about projection. Hence, select and the Projection Matrix (instead of the ModelView matrix) with:

```
glMatrixMode(GL_PROJECTION);
```

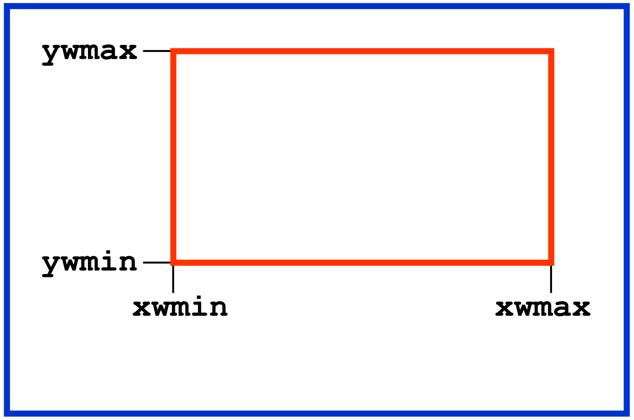
OpenGL 2D Viewing

Next, specify the 2D clipping window:

`gluOrtho2D(xwmin, xwmax, ywmin, ywmax);`

`xwmin, xwmax:` horizontal range, world coordinates

`ywmin, ywmax:` vertical range, world coordinates



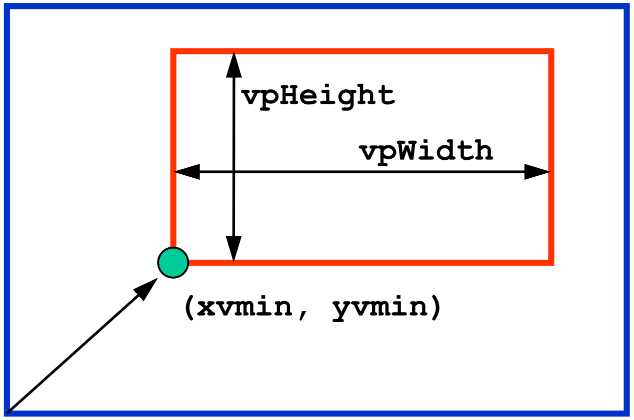
OpenGL 2D Viewing

Finally, specify the viewport:

`glViewport(xvmin, yvmin, vpWidth, vpHeight);`

`xvmin, yvmin:` coordinates lower left corner (in pixel coordinates);

`vpWidth, vpHeight:` width and height (in pixel coordinates);



OpenGL 2D Viewing

In short:

```
glMatrixMode(GL_PROJECTION);
gluOrtho2D(xwmin, xwmax, ywmin, ywmax);
glViewport(xvmin, yvmin, vpWidth, vpHeight);
```

To prevent distortion, make sure that:

```
(ywmax – ywmin)/(xwmax – xwmin) = vpWidth/vpHeight
```

OpenGL 2D viewing functions

- ***glMatrixMode(GL_PROJECTION);***
- ***glLoadIdentity();***
- ***glMatrixMode(GL_MODELVIEW);***
- ***gluOrtho2D(xwmin, xwmax, ywmin, ywmax);***
- ***glViewport(xvmin, yvmin, vpWidth, vpHeight);***
- ***glGetIntegerv (GL_VIEWPORT, vpArray);***
- ***glutInit(&argc,argv);***
- ***glutInitWindowPosition(10,10);***
- ***glutInitWindowSize(500,500);***
- ***glutCreateWindow("My First");***
- ***glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);***
- ***windowID = glutCreateWindow("My First");***
- ***glutDestroyWindow(windowID);***

OpenGL 2D viewing functions

- ***glutSetWindow(windowID);***
- ***currentWindowID = glutGetWindow();***
- ***glutReshapeWindow(width,height); //reset***
- ***glutFullScreen();***
- ***glutReshapeFunc(reshapeFunction);***
- ***glitIconifyWindow();***
- ***glutSetIconTitle("Icon Name");***
- ***glutSetWindowTitle("New Window Name");***
- ***glutSetWindow (windowID);***
- ***glutPopWindow ( );***
- ***glutSetWindow (windowID);***
- ***glutPushWindow ( );***
- ***glutHideWindow ( );***

OpenGL 2D viewing functions

- ***glutShowWindow ( );***
- ***glutCreateSubWindow (windowID, xBottomLeft, yBottomLeft,width, height);***
- ***glutDisplayFunc (pictureDescrip);***
- ***glutPostRedisplay ( );***
- ***glutMainLoop ( );***

Module-2

2D Geometric Transformations

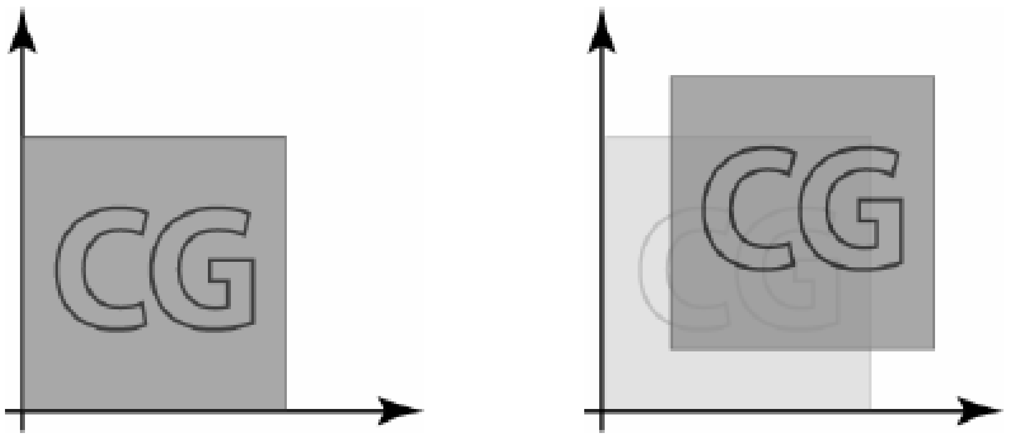
2D Transformations

"Transformations are the operations applied to geometrical description of an object to change its position, orientation, or size are called geometric transformations".

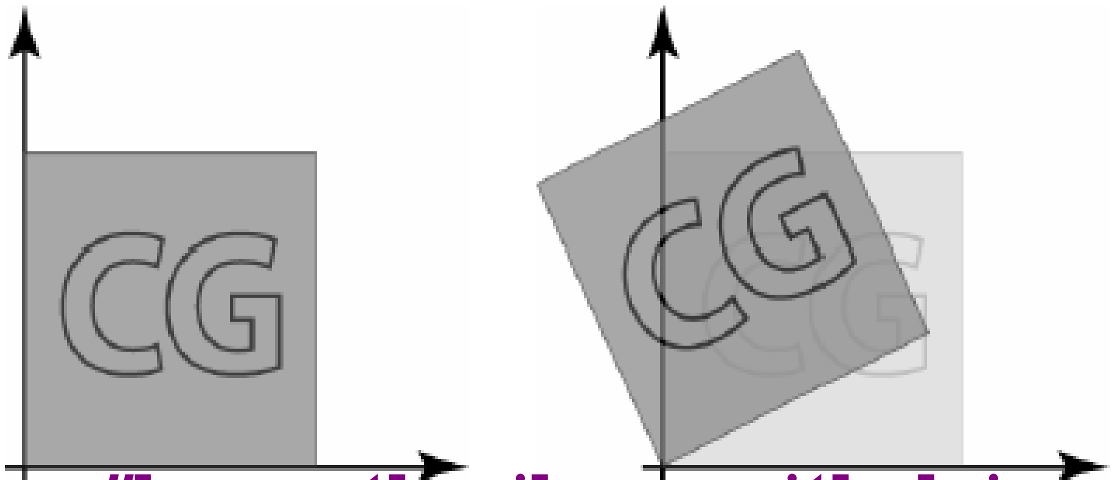
Why Transformations ?

"Transformations are needed to manipulate the initially created object and to display the modified object without having to redraw it."

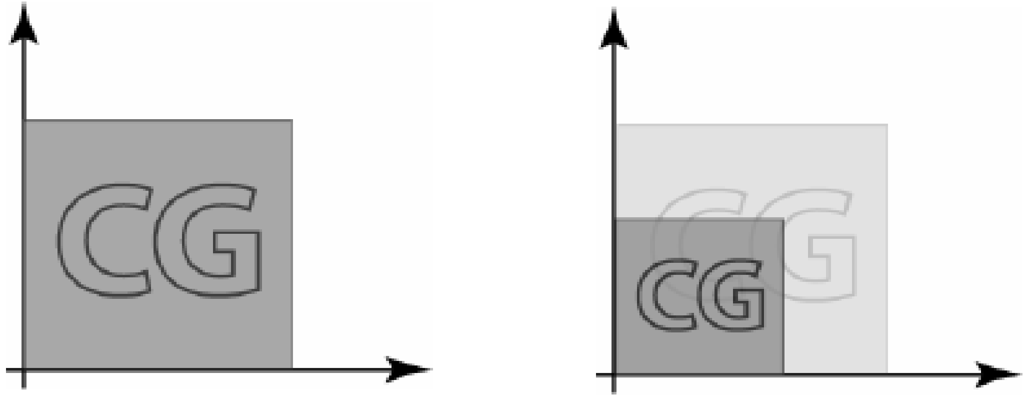
- Translation



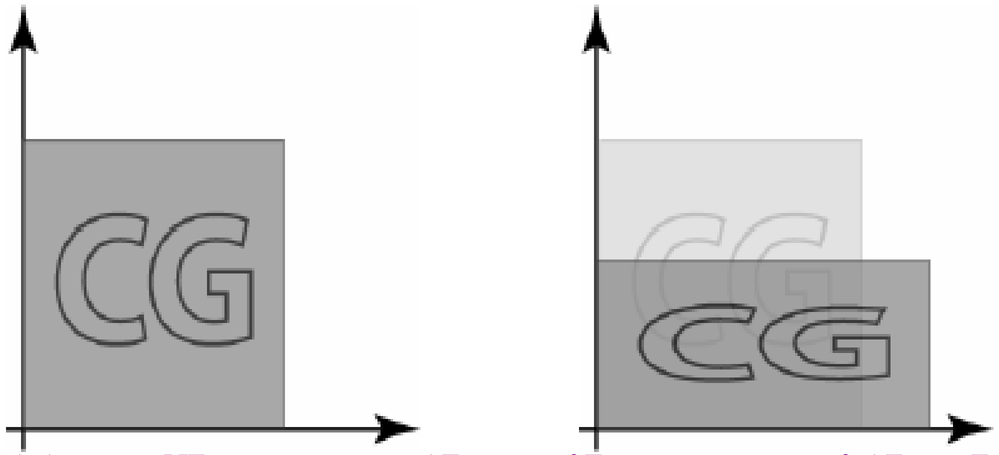
- Rotation



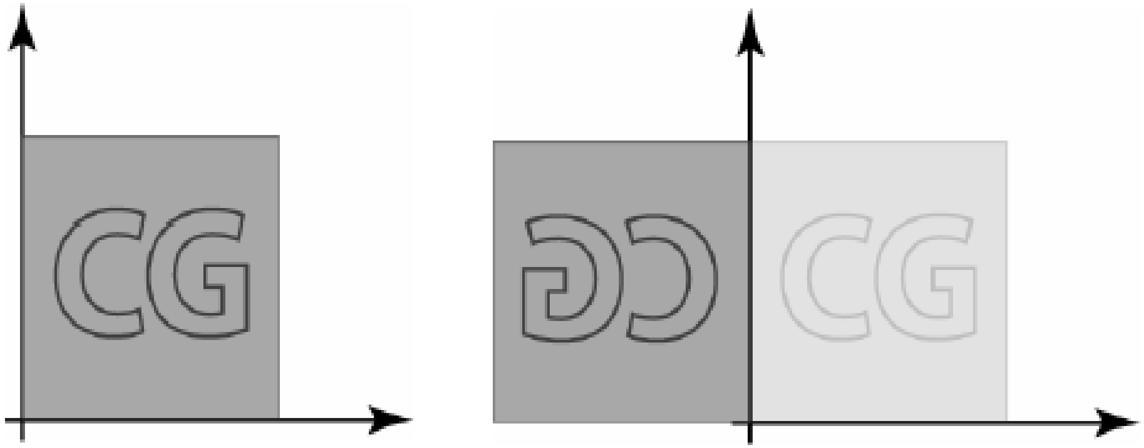
- Scaling
- Uniform Scaling



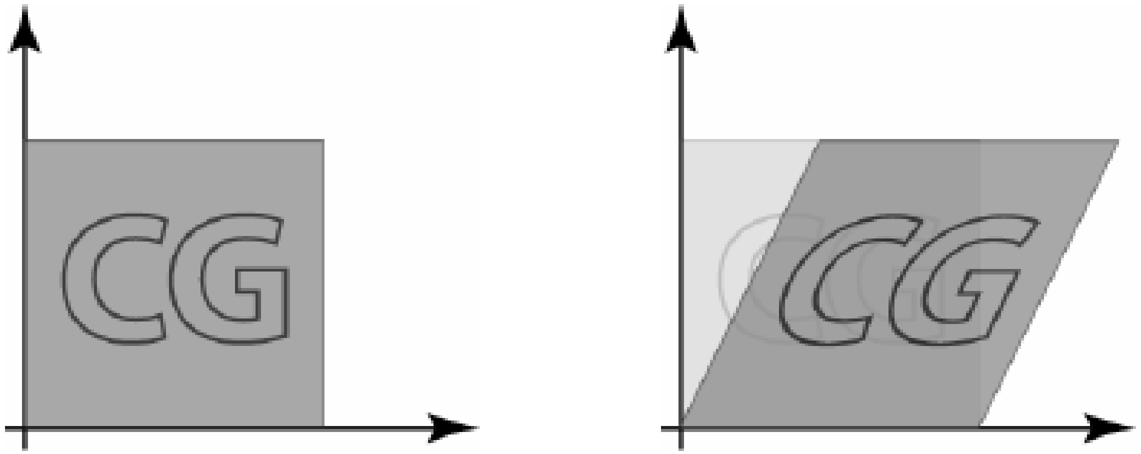
- Un-uniform Scaling



• Reflection



• Shear



Translation

- A translation moves all points in an object along the same straight-line path to new positions.
- The path is represented by a vector, called the translation or shift vector.
- We can write the components:

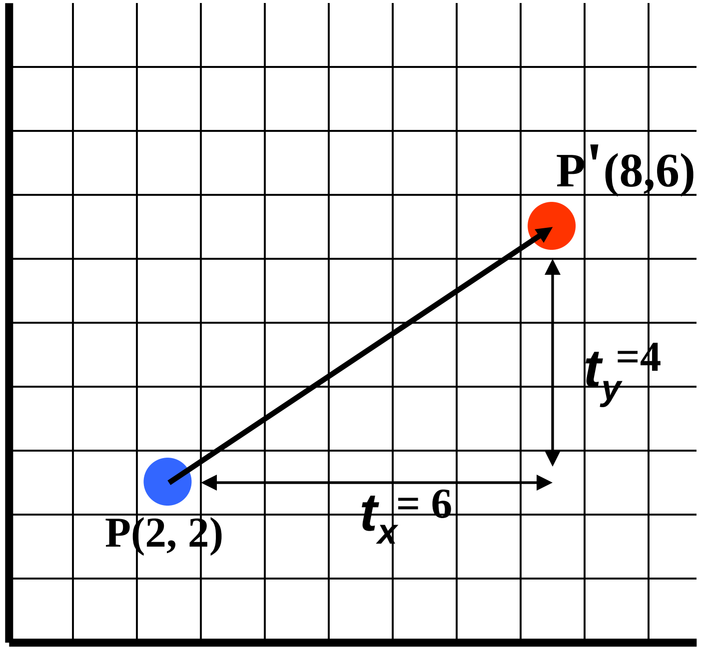
$$p'_x = p_x + t_x$$

$$p'_y = p_y + t_y$$

- or in matrix form:

$$P' = P + T$$

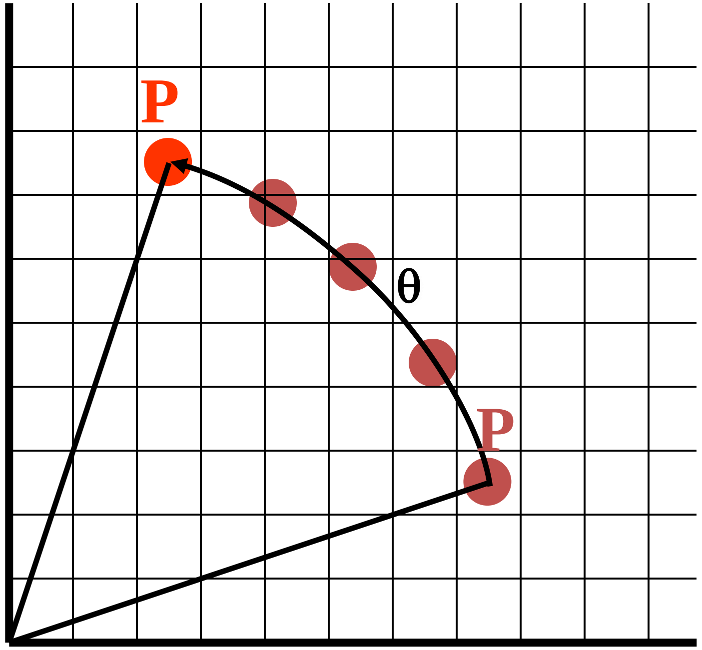
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$



Rotation

- A rotation repositions all points in an object along a circular path in the plane centered at the pivot point.

- First, we'll assume the pivot is at the origin.



Rotation

- Review Trigonometry

  => $\cos\phi = x/r$, $\sin\phi = y/r$

  - $x = r.\cos\phi$, $y = r.\sin\phi$

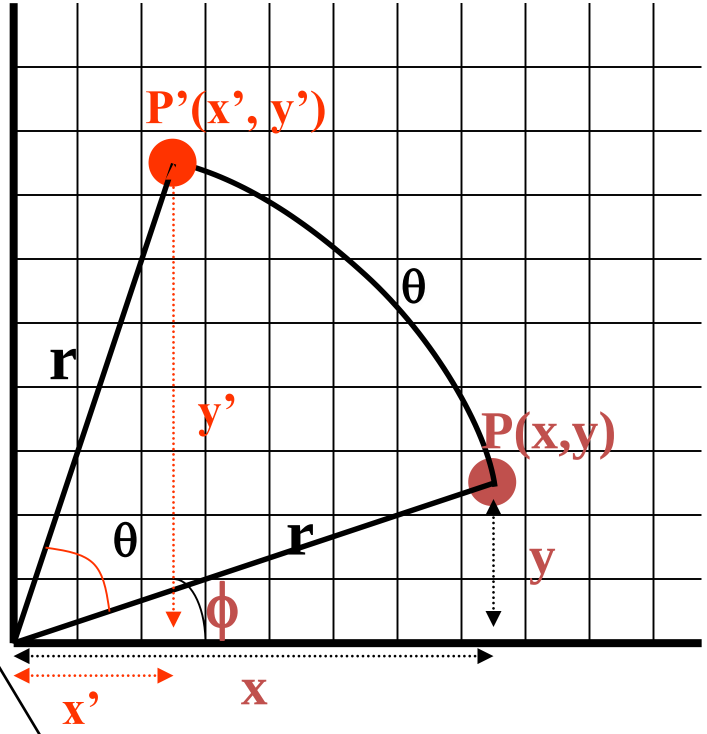
=> $\cos(\phi+\theta) = x'/r$

- $x' = r.\cos(\phi+\theta)$
- $x' = r.\cos\phi\cos\theta - r.\sin\phi\sin\theta$
- $x' = x.\cos\theta - y.\sin\theta$

=> $\sin(\phi+\theta) = y'/r$

$y' = r.\sin(\phi+\theta)$

- $y' = r.\cos\phi\sin\theta + r.\sin\phi\cos\theta$
- $y' = x.\sin\theta + y.\cos\theta$



Identity of Trigonometry

Rotation

- We can write the components:

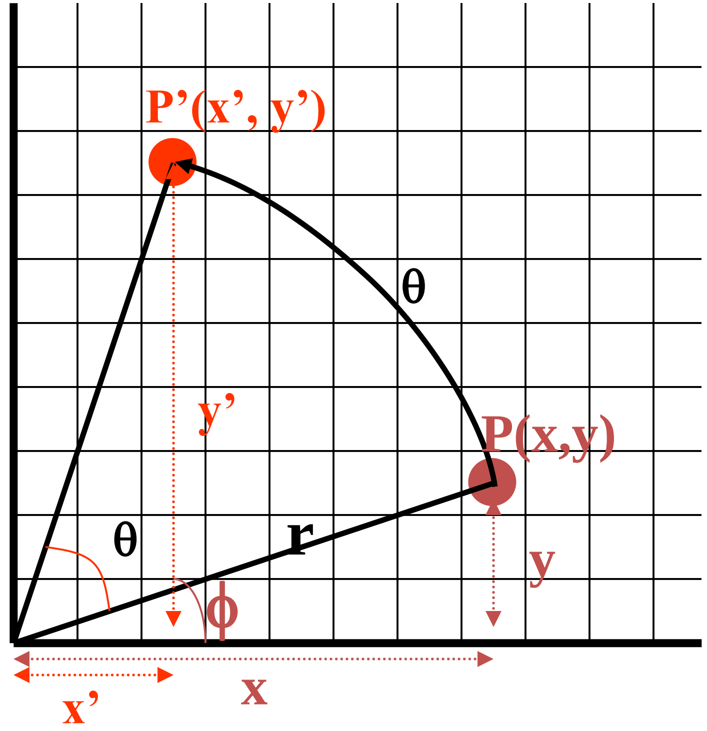
$$p'_x = p_x \cos\theta - p_y \sin\theta$$
$$p'_y = p_x \sin\theta + p_y \cos\theta$$

- or in matrix form:

P' = R • P

- θ can be clockwise (-ve) or counterclockwise (+ve as our example).
- Rotation matrix

$$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$



Scaling

- Scaling changes the size of an object and involves two scale factors, S_x and S_y for the x- and y- coordinates respectively.
- Scales are about the origin.
- We can write the components:

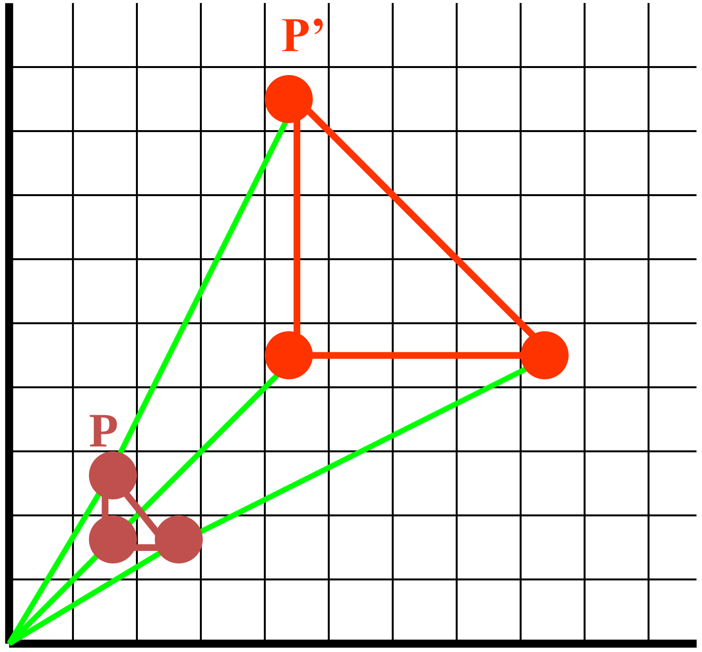
$$p'_x = s_x \bullet p_x$$
$$p'_y = s_y \bullet p_y$$

or in matrix form:

$$P' = S \bullet P$$

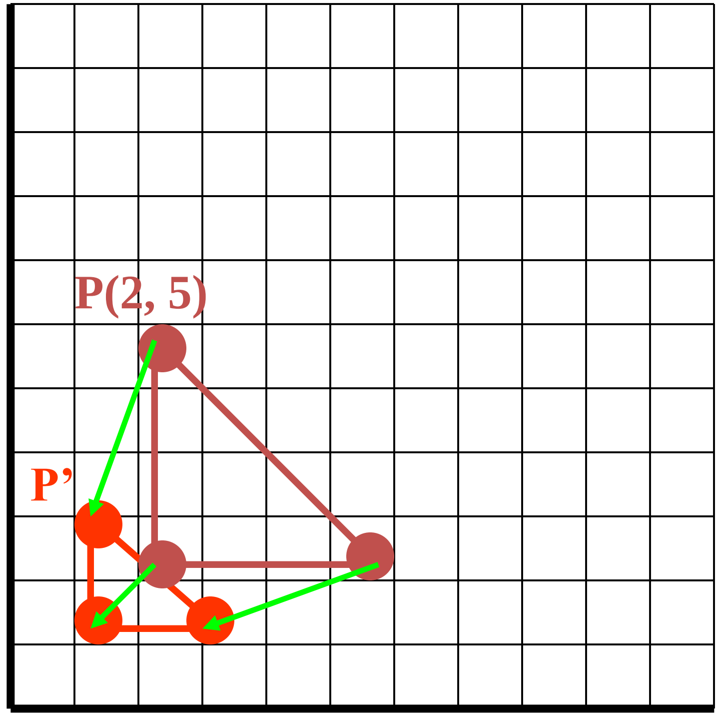
Scale matrix as:

$$S = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix}$$



Scaling

- If the scale factors are in between 0 and 1:----
- ➔ the points will be moved closer to the origin
- ➔ the object will be smaller.



- Example :
  - P(2, 5), $S_x = 0.5$, $S_y = 0.5$

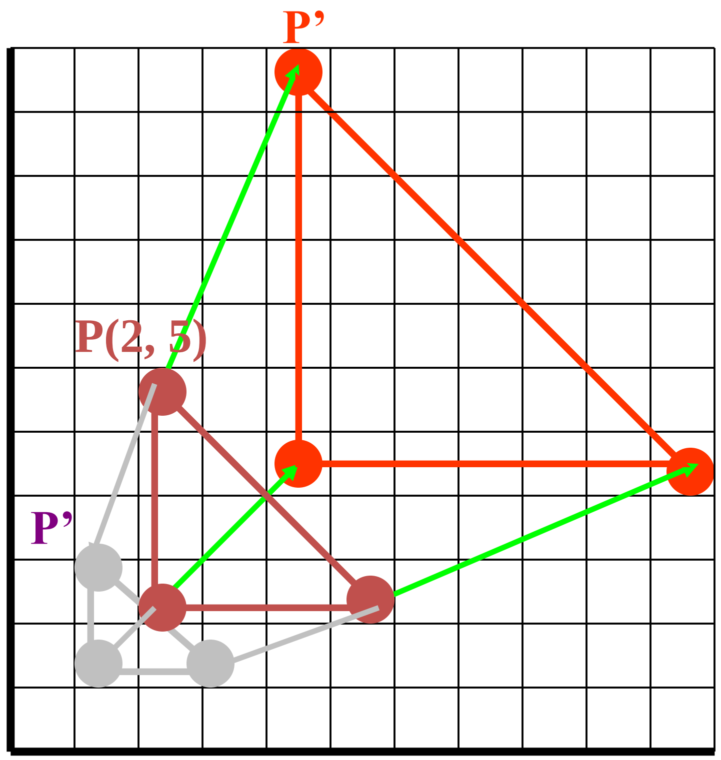
Scaling

- **If the scale factors are in between 0 and 1 ➔ the points will be moved closer to the origin ➔ the object will be smaller.**

- **Example :**
  - **P(2, 5), S_x = 0.5, S_y = 0.5**

- **If the scale factors are larger than 1 ➔ the points will be moved away from the origin ➔ the object will be larger.**

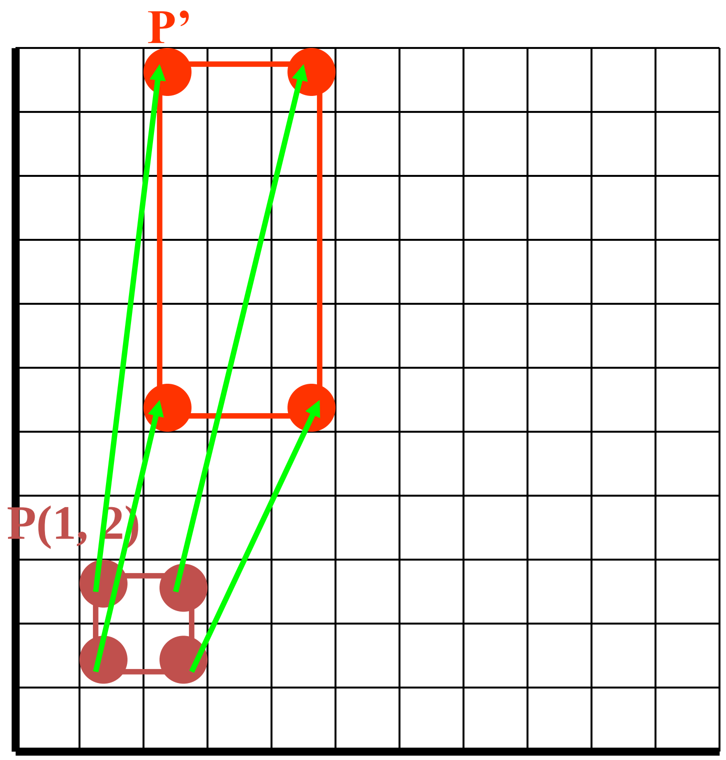
- **Example :**
  - **P(2, 5), S_x = 2, S_y = 2**



Scaling

- If the scale factors are the same, $S_x = S_y$ ➔ uniform scaling
- Only change in size (as previous example)

- If $S_x \neq S_y$ ➔ differential scaling.
- Change in size and shape
- Example : square → rectangle
  - P(1, 3), $S_x = 2$, $S_y = 5$



Matrix Representations & Homogenous Coordinates

$$P' = P + T$$

$$P' = S \bullet P$$

$$P' = R \bullet P$$

$$\text{Translation } P' = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

$$\text{Rotation } P' = \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\text{Scaling } P' = \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Combining above equations, we can say that

$$P' = M_1 \bullet P + M_2$$

Using homogenous co-ordinates, the transformations could be combined easily. Here we reformulate equation to eliminate matrix addition.

In homogenous co-ordinate system, we combine multiplicative and translational terms by expanding the **2 × 2 matrix** representation to **3 × 3 matrices**. Also expand the matrix representation for co-ordinate position

We represent each Cartesian co-ordinate(x, y) with homogeneous co-ordinate (x_h, y_h, h) where $x = x_h/h$, $y = y_h/h$

$$(h^*x, h^*y, h)$$

$$\text{set } h = 1$$

$$(x, y, 1)$$

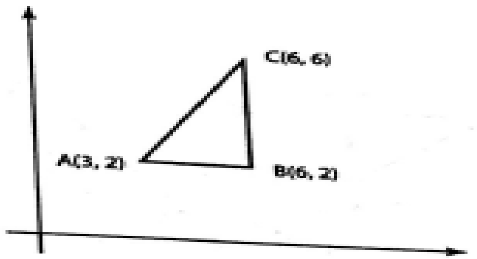
Homogenous co-ordinates representation for translation, scaling and rotation are as follows:

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

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

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Problem: Rotate the given Triangle by 90 degrees about the origin



rigin by 90°

Solution

Applying homogenous co-ordinate system for rotation,

For co-ordinate A (3, 2),

$$\begin{pmatrix}x'\\y'\\1\end{pmatrix} = \begin{pmatrix}\cos(90) & -\sin(90) & 0\\ \sin(90) & \cos(90) & 0\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}3\\2\\1\end{pmatrix} = \begin{pmatrix}0 & -1 & 0\\ 1 & 0 & 0\\ 0 & 0 & 1\end{pmatrix}\cdot\begin{pmatrix}3\\2\\1\end{pmatrix}$$

$$A(x', y', 1) = (-2,3,1)$$

For co-ordinate B (6, 2),

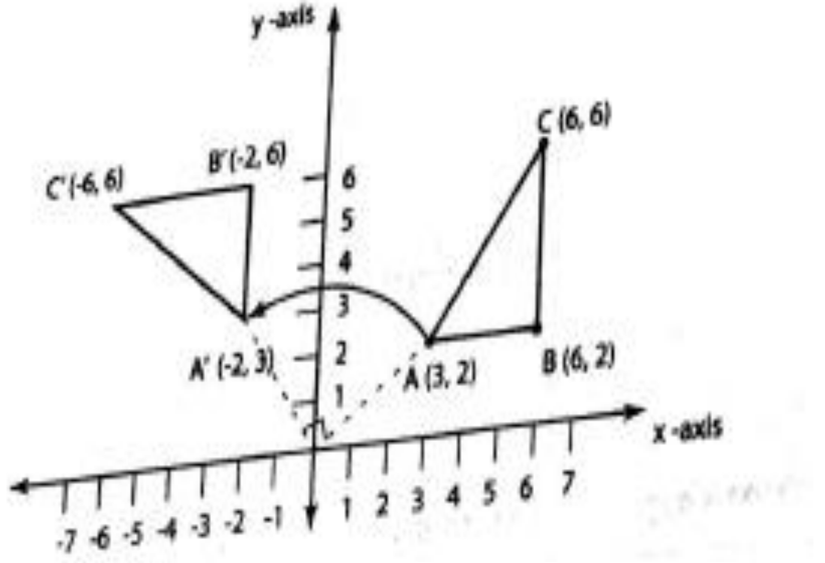
$$\begin{pmatrix}x'\\y'\\1\end{pmatrix} = \begin{pmatrix}\cos(90) & -\sin(90) & 0\\ \sin(90) & \cos(90) & 0\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}6\\2\\1\end{pmatrix} = \begin{pmatrix}0 & -1 & 0\\ 1 & 0 & 0\\ 0 & 0 & 1\end{pmatrix}\cdot\begin{pmatrix}6\\2\\1\end{pmatrix}$$

$$B(x', y', 1) = (-2,6,1)$$

For co-ordinate C (6, 6),

$$\begin{pmatrix}x'\\y'\\1\end{pmatrix} = \begin{pmatrix}\cos(90) & -\sin(90) & 0\\ \sin(90) & \cos(90) & 0\\ 0 & 0 & 1\end{pmatrix}\begin{pmatrix}6\\6\\1\end{pmatrix} = \begin{pmatrix}0 & -1 & 0\\ 1 & 0 & 0\\ 0 & 0 & 1\end{pmatrix}\cdot\begin{pmatrix}6\\6\\1\end{pmatrix}$$

$$C(x', y', 1) = (-6,6,1)$$



Triangle rotated about the origin by 90°

Problem: Prove that successive translations are additive

2. Prove that successive translations are additive

If a point P is translated by $T(tx_1, ty_1)$ to P′ and then translated by $T(tx_2, ty_2)$ to P″

$$P' = T(tx_1, ty_1)^*P \quad (3.6)$$

$$P'' = T(tx_2, ty_2)^*P' \quad (3.7)$$

Substituting equation (3.6) into (3.7), we obtain

$$P'' = T(tx_2, ty_2)^* (T(tx_1, ty_1)^* P)$$

$$= (T(tx_2, ty_2)^* T(tx_1, ty_1))^*P$$

$$p^* = \begin{bmatrix} 1 & 0 & tx_2 \\ 0 & 1 & ty_2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & tx_1 \\ 0 & 1 & ty_1 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$p^* = \begin{bmatrix} 1 & 0 & tx_1 + tx_2 \\ 0 & 1 & ty_1 + ty_2 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

The matrix product $(T(Tx_2, ty_2)^*T(tx_1, ty_1))$ is the net translation is indeed $T(tx_1 + tx_2, ty_1 + ty_2)$

Problem: Prove that successive scaling is multiplicative

$$P' = S(sx_1, sy_1)^* \, P \tag{3.8}$$

$$P'' = S(sx_2, sy_2)^* \, P' \tag{3.9}$$

Substituting equation (3.8) in (3.9)

$$P'' = S(sx_2, sy_2)^* \, (S(sx_1, sy_1)^* \, P)$$

$$= (S(sx_2, sy_2)^* \, S(sx_1, sy_1))^* \, P$$

The matrix product $S(sx_2, sy_2)^* \, S(sx_1, sy_1)$ is the net scaling transformations.Thus, scaling is indeed multiplicative.

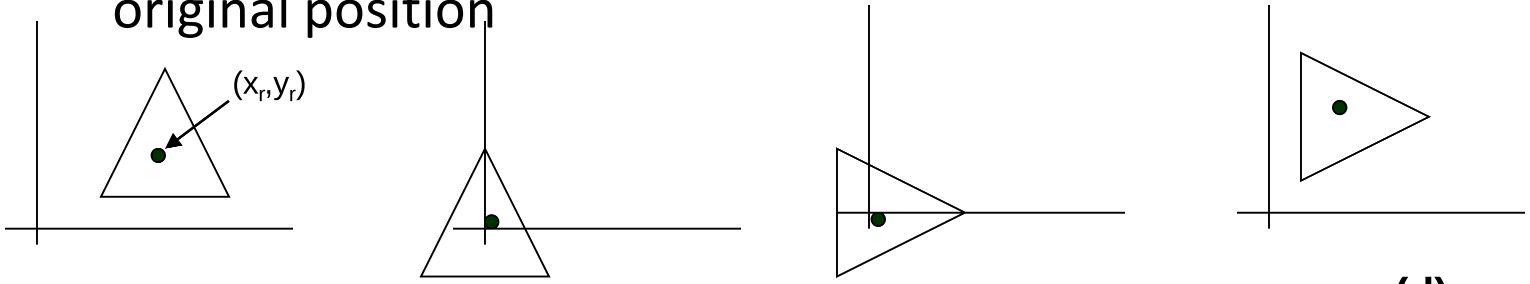
$$P^* = \begin{bmatrix} sx_2 & 0 & 0 \\ 0 & sy_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} sx_1 & 0 & 0 \\ 0 & sy_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$P^* = \begin{bmatrix} sx_1 \cdot sx_2 & 0 & 0 \\ 0 & sy_1 \cdot sy_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- Similarly successive rotations are additive.

General pivot point rotation

- Translate the object so that pivot-position is moved to the coordinate origin
- Rotate the object about the coordinate origin
- Translate the object so that the pivot point is returned to its original position



(a) Original Position of Object and pivot point

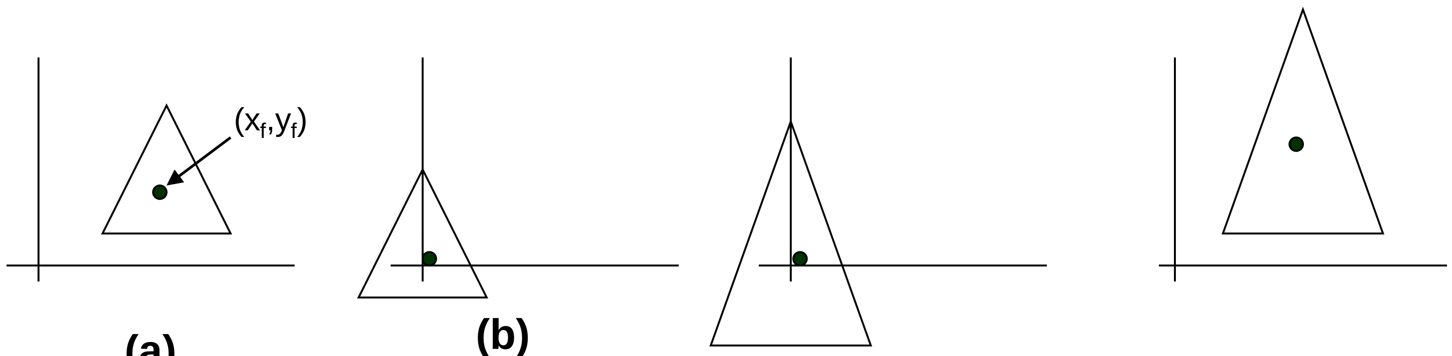
(b) Translation of object so that pivot point (x_r,y_r) is at origin

(c) Rotation was about origin

(d) Translation of the object so that the pivot point is returned to position (x_r,y_r)

General fixed point scaling

- Translate object so that the fixed point coincides with the coordinate origin
- Scale the object with respect to the coordinate origin
- Use the inverse translation of step 1 to return the object to its original position



(a) Original Position of Object and Fixed point

(b) Translation of object so that fixed point (x_f,y_f)is at origin

(c) scaling was about origin

(d) Translation of the object so that the Fixed point is returned to position (x_f,y_f)

Composite Transformations

(A) Translations

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 as: -

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

Where P and P' are represented as homogeneous-coordinate column vectors. We can verify this result by calculating the matrix product for the two associative groupings. Also, the composite transformation matrix for this sequence of transformations is: -

$$\begin{vmatrix} 1 & 0 & t_{x2} \\ 0 & 1 & t_{y2} \\ 0 & 0 & 1 \end{vmatrix} \cdot \begin{vmatrix} 1 & 0 & t_{x1} \\ 0 & 1 & t_{y1} \\ 0 & 0 & 1 \end{vmatrix} = \begin{vmatrix} 1 & 0 & t_{x1}+t_{x2} \\ 0 & 1 & t_{y1}+t_{y2} \\ 0 & 0 & 1 \end{vmatrix}$$

Or, $\mathbf{T(t_{x2},t_{y2}) \cdot T(t_{x1},t_{y1}) = T(t_{x1}+t_{x2},\ t_{y1}+t_{y2})}$

Which demonstrate that two successive translations are additive.

(B) Rotations

Two successive rotations applied to point P produce the transformed position: -

$$P' = R(\Theta_2) \,.\, \{R(\Theta_1) \,.\, P\}$$
$$= \{R(\Theta_2) \,.\, R(\Theta_1)\} \,.\, P$$

By multiplication the two rotation matrices, we can verify that two successive rotations are additive:

$$\mathbf{R(\Theta_2) \,.\, R(\Theta_1) = R\,(\Theta_1 + \Theta_2)}$$

So that the final rotated coordinates can be calculated with the composite rotation matrix as: -

$$\mathbf{P' = R(\Theta_1 + \Theta_2) \,.\, P}$$

(C) Scaling

Concatenating transformation matrices for two successive scaling operations produces the following composite scaling matrix: -

$$\begin{vmatrix} S_{x2} & 0 & 0 \\ 0 & S_{y2} & 0 \\ 0 & 0 & 1 \end{vmatrix} \cdot \begin{vmatrix} S_{x1} & 0 & 0 \\ 0 & S_{y1} & 0 \\ 0 & 0 & 1 \end{vmatrix} = \begin{vmatrix} S_{x1} \cdot S_{x2} & 0 & 0 \\ 0 & S_{y1} \cdot S_{y2} & 0 \\ 0 & 0 & 1 \end{vmatrix}$$

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

The resulting matrix in this case indicates that successive scaling operations are multiplicative.

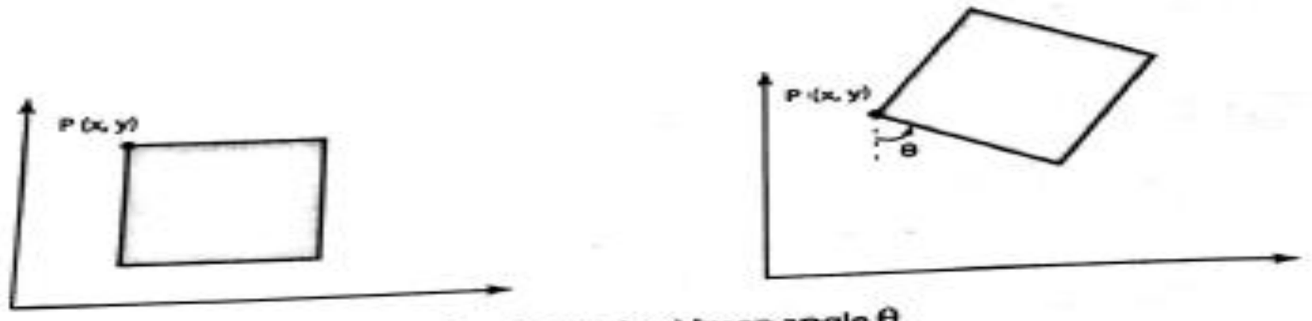
2D Composition Problems

THEORY

Rotate an object about an arbitrary point P_f:

To rotate about P_f, we need a sequence of 3 fundamental transformations

- Translate such that P_f is at the origin
- Rotate
- Translate such that the point at the origin returns to P_f



e the square about the fixed point (x, y) by an angle θ

The net transformation is

$T(x_f, y_f) \cdot R(\theta) \cdot T(-x_f, -y_f)$

$$= \begin{bmatrix} 1 & 0 & x_r \\ 0 & 1 & y_r \\ 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} 1 & 0 & -x_r \\ 0 & 1 & -y_r \\ 0 & 0 & 1 \end{bmatrix}$$

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

PROBLEM 1

problem

4. *Draw a polygon ABC. A(3, 2), B(6, 2) and C(6, 6) rotate it in anticlockwise direction by 90 degree by keeping a point A(3, 2) fixed.*

Sequence of transformations applied to the polygon ABC are as follows:

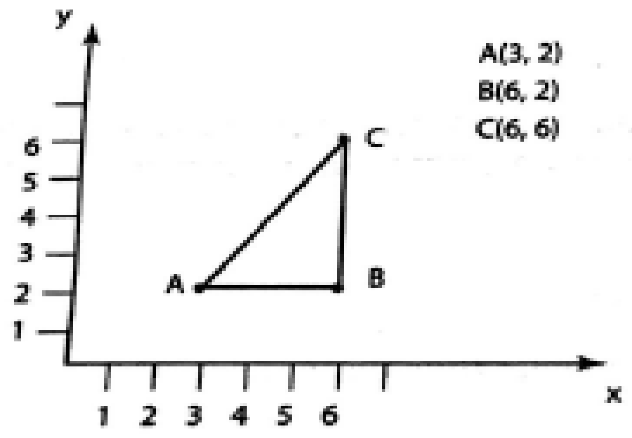


Figure 3.10 Origin position of the polygon

Step 1

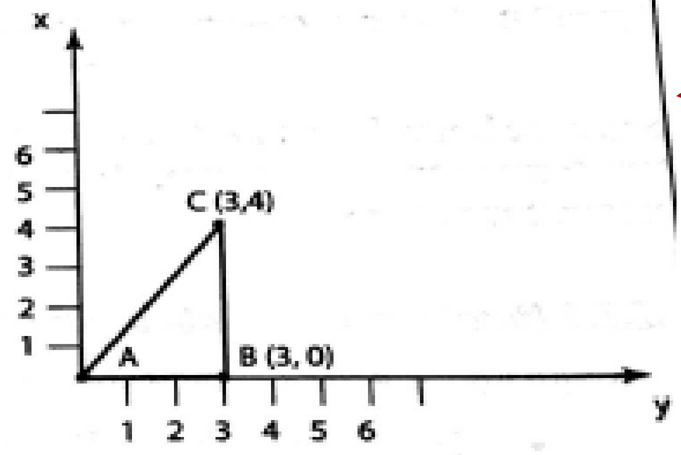


Figure 3.11 Translate the polygon to the origin by translation factor $t_x = -3, t_y = -2$

Step 2

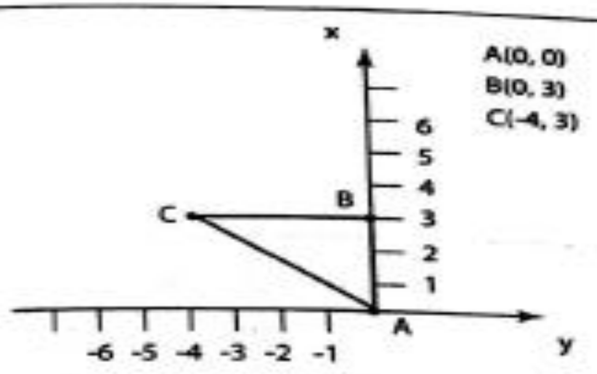


Figure 3.12 Rotate the polygon anti-clockwise by 90° degrees

Step 3

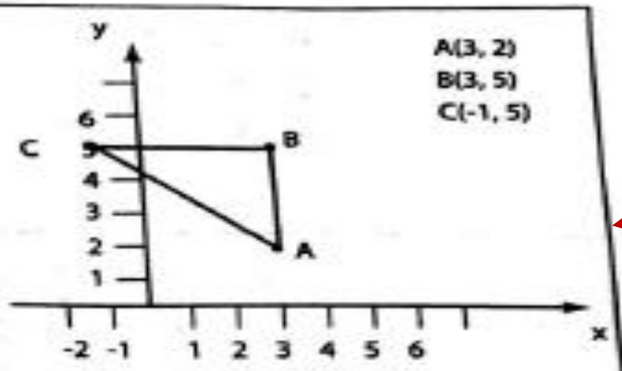


Figure 3.13 Translate the polygon by translation factor $t_x = 3, t_y = 2$

The transformations shown in figure 3.10 to 3.13, could be done using homogenous co-ordinates as follows:

$$P' = T(x, y)^{\bullet} R(\theta)^{\bullet} T(-x, -y)^{\bullet} P(x, y)$$

θ is positive because anti-clockwise rotation

$$P' = T(3, 2)^{\bullet} R(90^{o})^{\bullet} T(-3, -2)^{\bullet} P(x, y)$$

$$= \begin{bmatrix} 1 & 0 & x_t \\ 0 & 1 & y_t \\ 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} 1 & 0 & -x_t \\ 0 & 1 & -y_t \\ 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 3 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} \cos 90 & -\sin 90 & 0 \\ \sin 90 & \cos 90 & 0 \\ 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} 1 & 0 & -3 \\ 0 & 1 & -2 \\ 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 3 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} 1 & 0 & -3 \\ 0 & 1 & -2 \\ 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 3 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} 0 & -1 & 2 \\ 1 & 0 & -3 \\ 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & -1 & 5 \\ 1 & 0 & -1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$P'_B = \begin{bmatrix} 0 & -1 & 5 \\ 1 & 0 & -1 \\ 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} 6 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 3 \\ 5 \\ 1 \end{bmatrix}$	$P'_C = \begin{bmatrix} 0 & -1 & 5 \\ 1 & 0 & -1 \\ 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} 6 \\ 6 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 \\ 5 \\ 1 \end{bmatrix}$
Solution: $P'_A(3,2)$ $P'_B(3,5)$ $P'_C(-1,5)$	

Difference between rotation of a triangle about the origin by 90°, and rotation about a fixed point A(3, 2) is shown in fig 3.13a

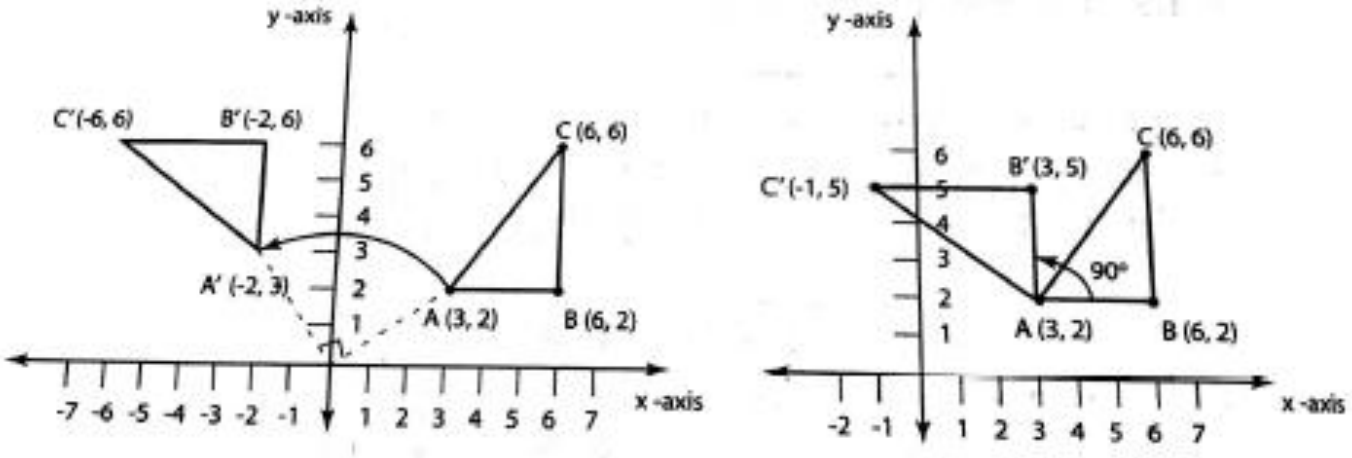


Figure 3.13a Difference between rotation about the origin and fixed point rotation

3.2.1.2 *Scale an object about an arbitrary point P_f*

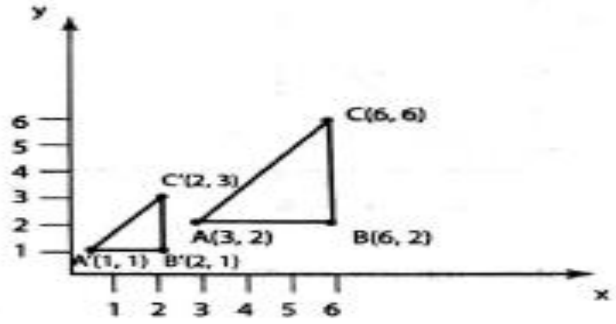
To scale an object about an arbitrary point P_f the following three steps are required:

- Translate such that P_f goes to origin
- Scale
- Translate back to P_f

Composition of these transformation is: $T(x_f, y_f)^{\bullet}\ S(s_x, s_y)^{\bullet}\ T(-x_f, -y_f)$

5. *Scale the given triangle A(3, 2) B(6, 2) C(6, 6) using the scaling factors $S_x = 1/3$ $S_y = 1/2$ about the origin [figure 3.14].*

PROBLEM 2



gure 3.14 Scale the triangle ABC about the origin by $s_x = 1/3$ and $s_y = 1/2$

caled with respect to origin

$$P' = S(s_x, s_y) \cdot P(x, y)$$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$P'_A = \begin{bmatrix} 1/3 & 0 & 0 \\ 0 & 1/2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} 3 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

$$P'_B = \begin{bmatrix} 1/3 & 0 & 0 \\ 0 & 1/2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} 6 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix}$$

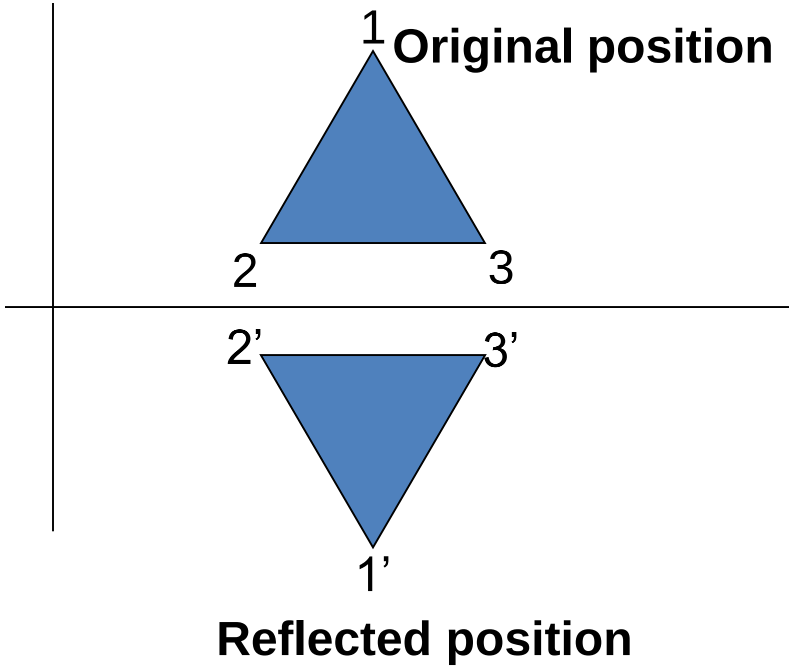
$$P'_C = \begin{bmatrix} 1/3 & 0 & 0 \\ 0 & 1/2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} 6 \\ 6 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 3 \\ 1 \end{bmatrix}$$

Solution

$P'_A(1, 1)$
$P'_B(2, 1)$
$P'_C(2, 3)$

Other transformations

- **Reflection** is a transformation that produces a mirror image of an object. It is obtained by rotating the object by 180 deg about the reflection axis

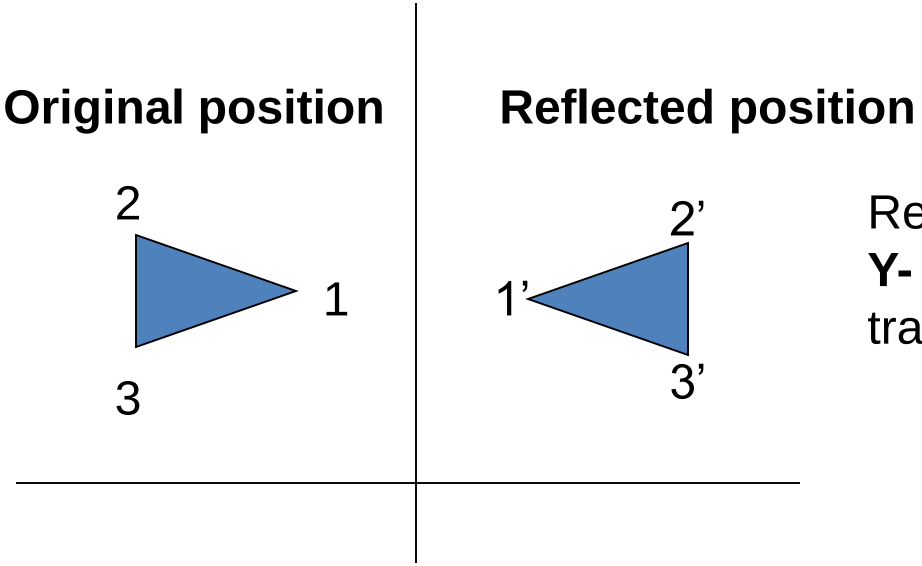


Reflection about the line y=0, the **X- axis** , is accomplished with the transformation matrix

$$\begin{vmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{vmatrix}$$

https://hemanthrajhemu.github.io

Reflection

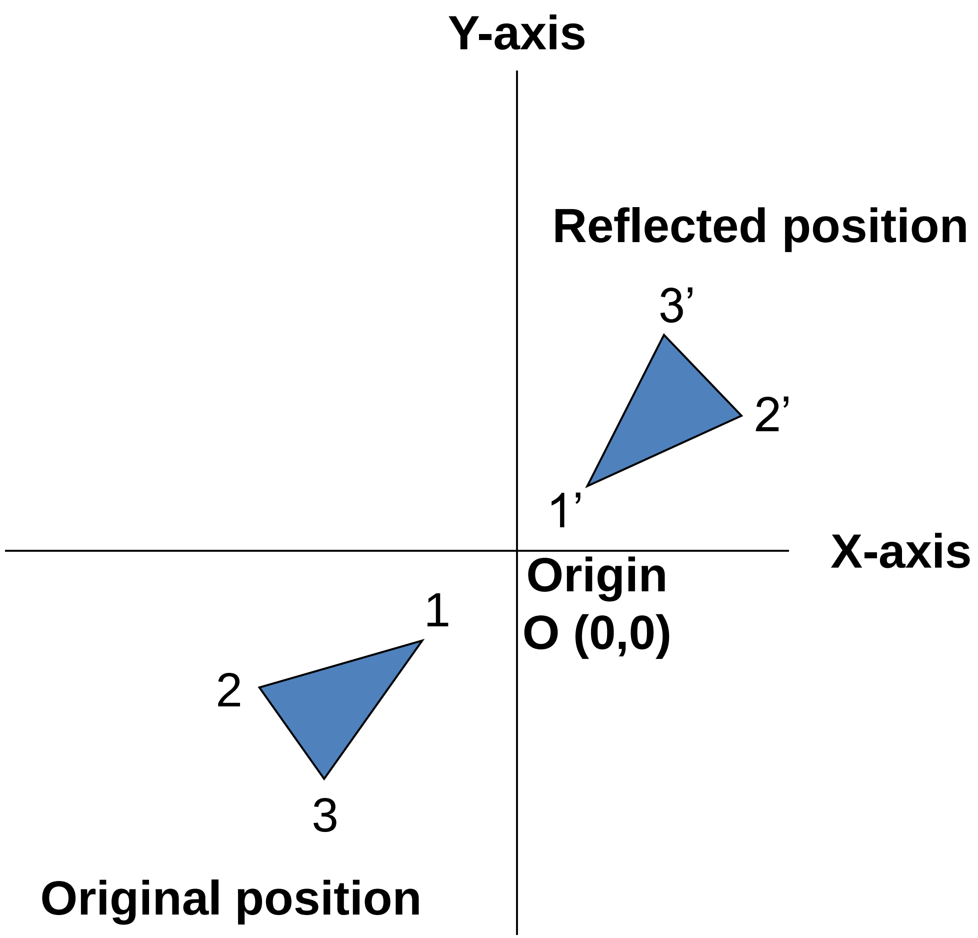


Reflection about the line x=0, the **Y- axis** , is accomplished with the transformation matrix

$$\begin{vmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{vmatrix}$$

Reflection

Reflection of an object relative to an axis perpendicular to the xy plane and passing through the coordinate origin



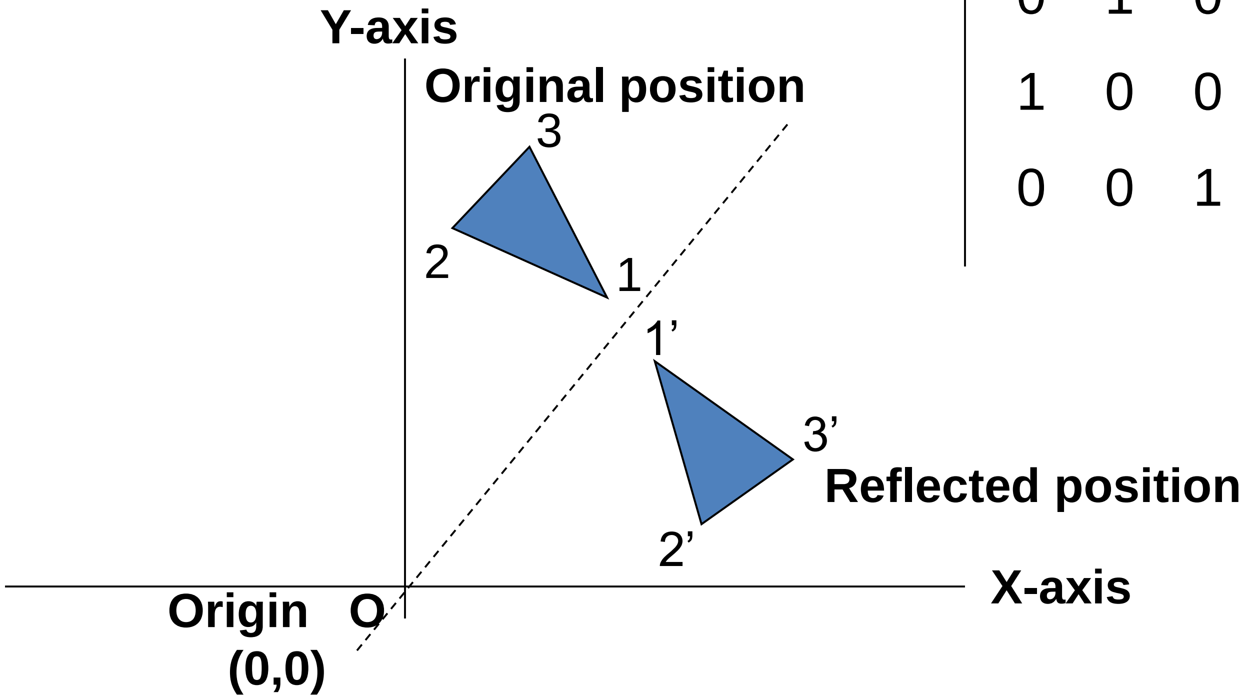
$$\begin{vmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{vmatrix}$$

The above reflection matrix is the rotation matrix with angle=180 degree.

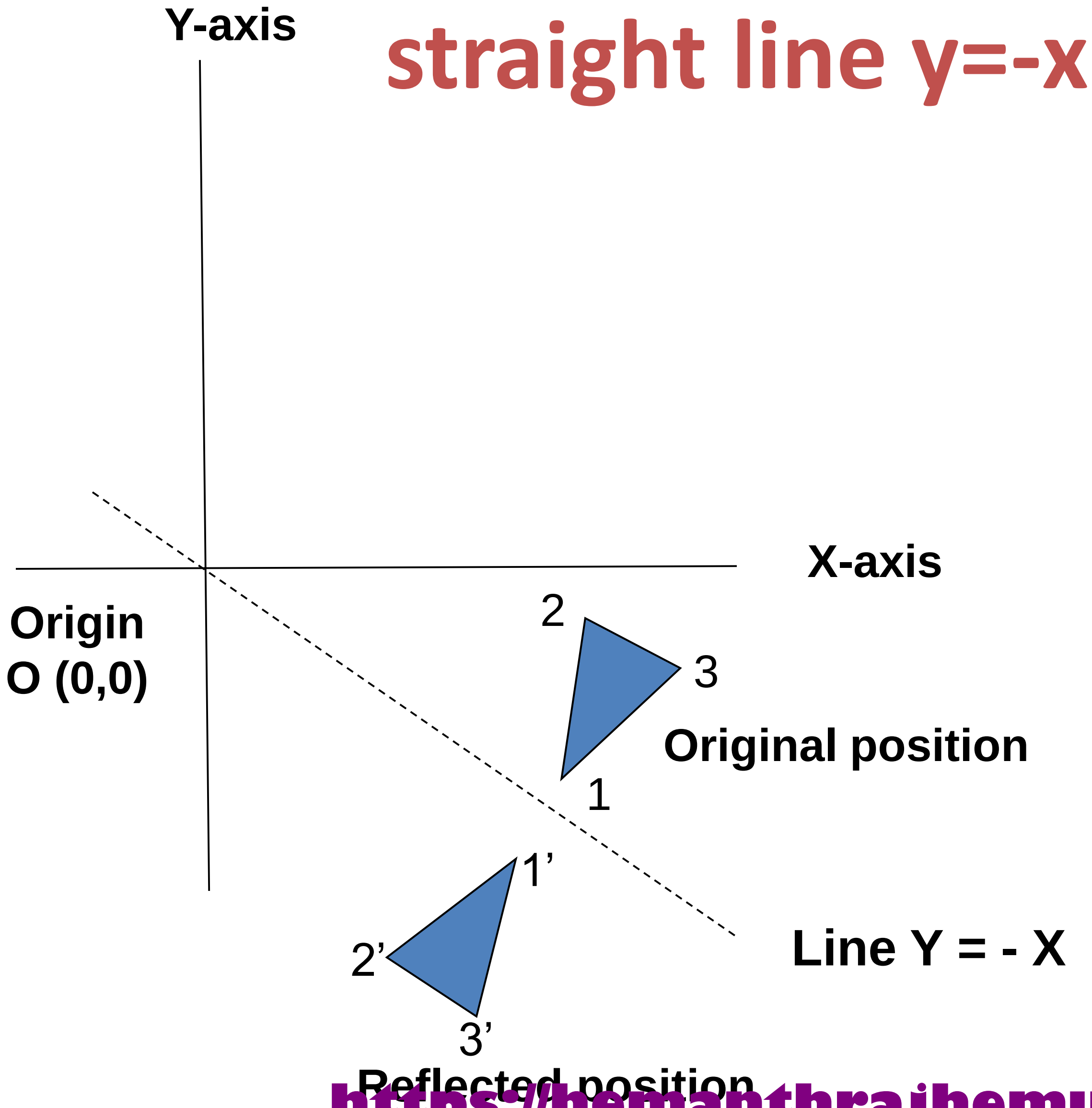
This can be generalized to any reflection point in the xy plane. This reflection is the same as a 180 degree rotation in the xy plane using the reflection point as the pivot point.

Reflection of an object w.r.t the straight line y=x

$$\begin{vmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{vmatrix}$$

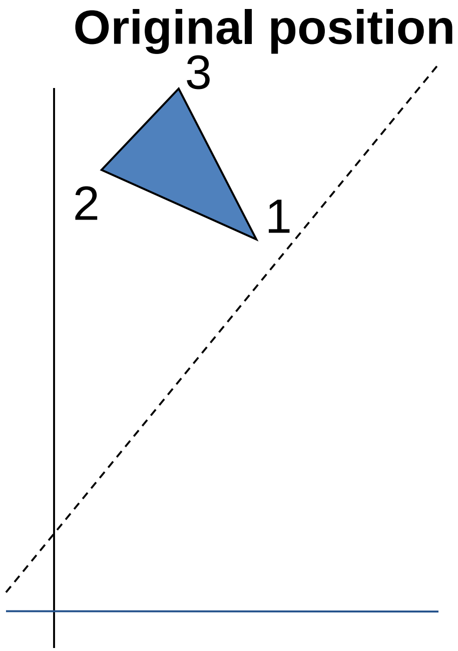


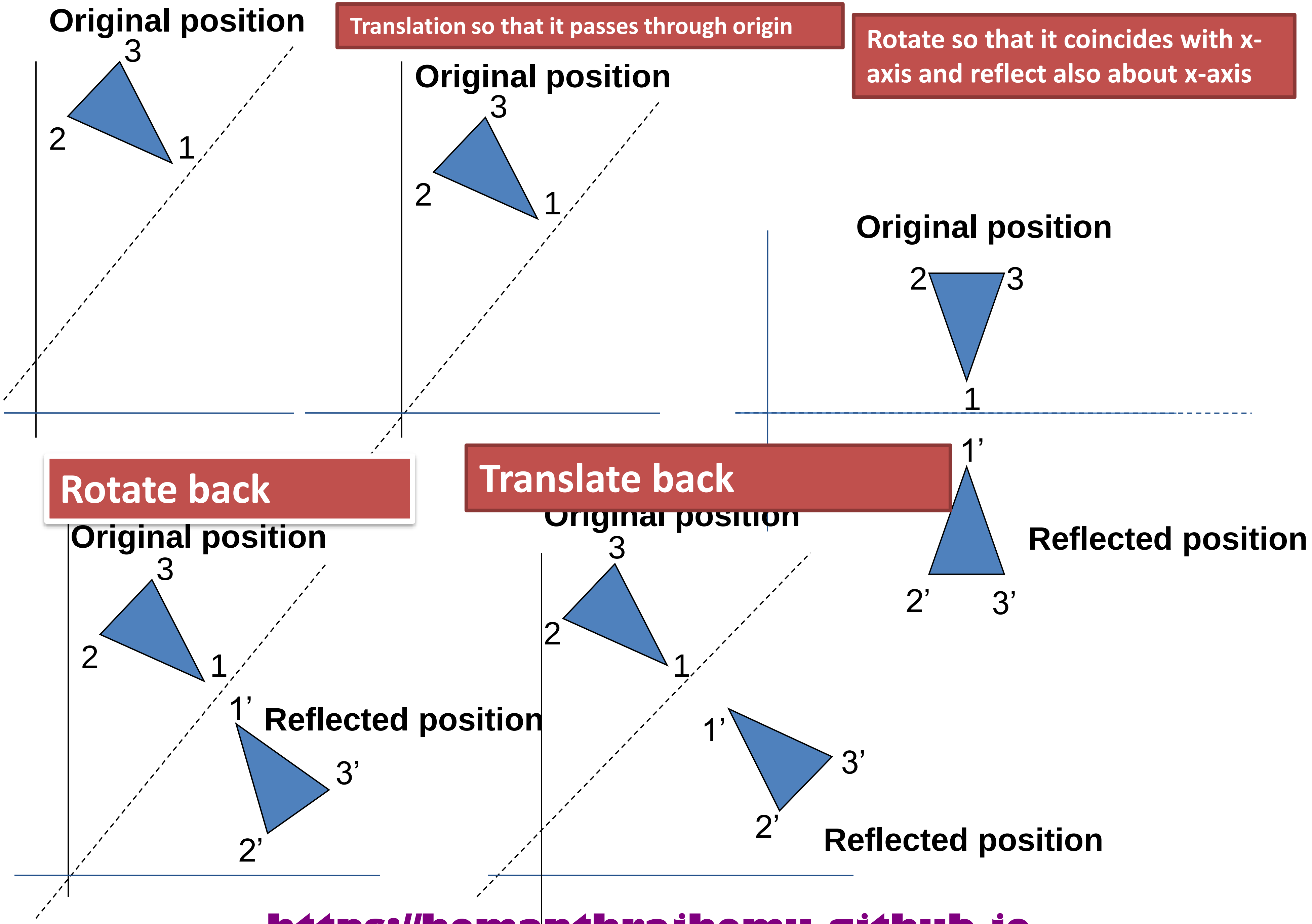
Reflection of an object w.r.t the straight line y=-x



$$\begin{vmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{vmatrix}$$

Reflection of an arbitrary axis
y=mx+b

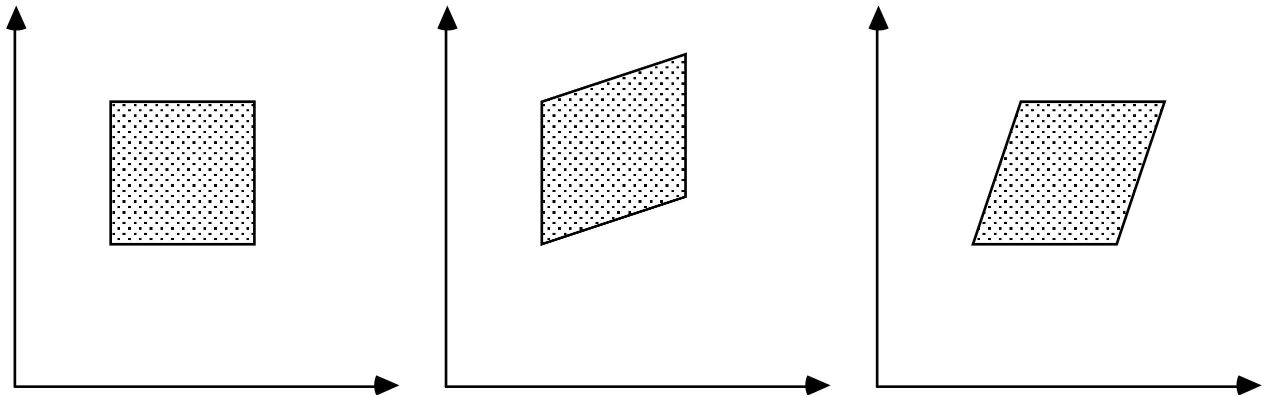



Original position
Translation so that it passes through origin
Rotate so that it coincides with x-axis and reflect also about x-axis
Original position
3
2
1
Original position
3
2
1
Original position
2
3
1
1’
2’
3’
Reflected position
Rotate back
Original position
3
2
1
1’
Reflected position
3’
2’
Translate back
Original position
3
2
1
1’
3’
2’
Reflected position

Shear Transformations

- Shear is 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
- Two common shearing transformations are those that shift coordinate x values and those that shift y values

Shears

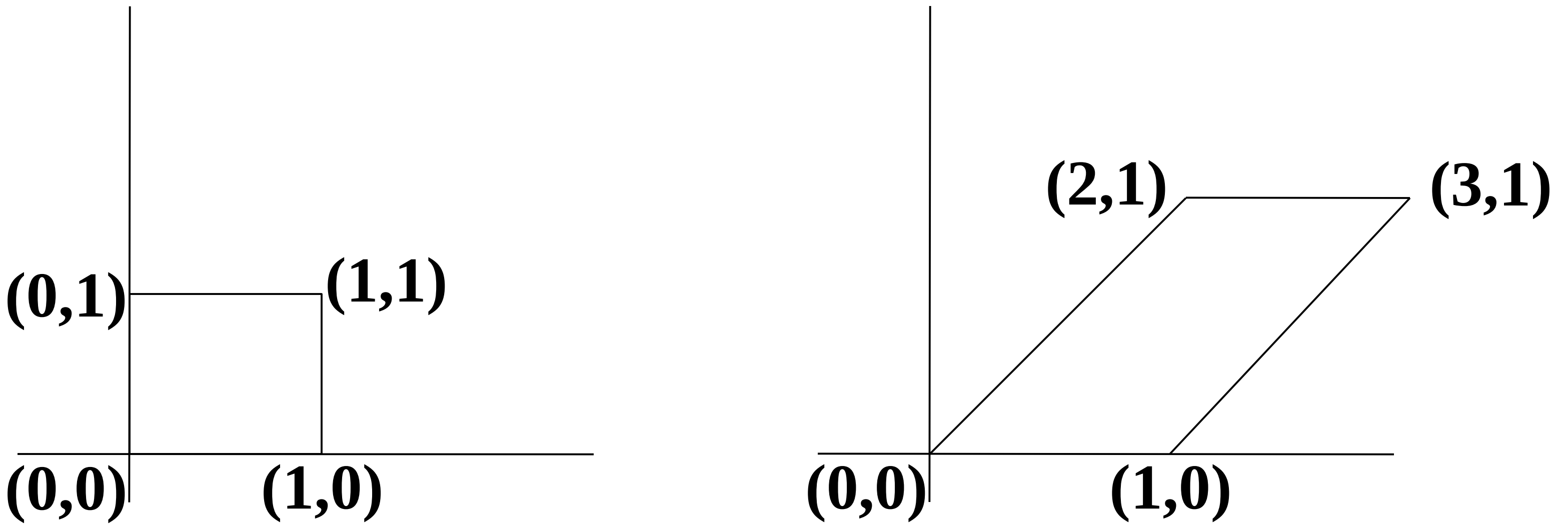


Original Data **y Shear** **x Shear**

$$\begin{bmatrix} 1 & 0 & 0 \\ sh_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} 1 & sh_x & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

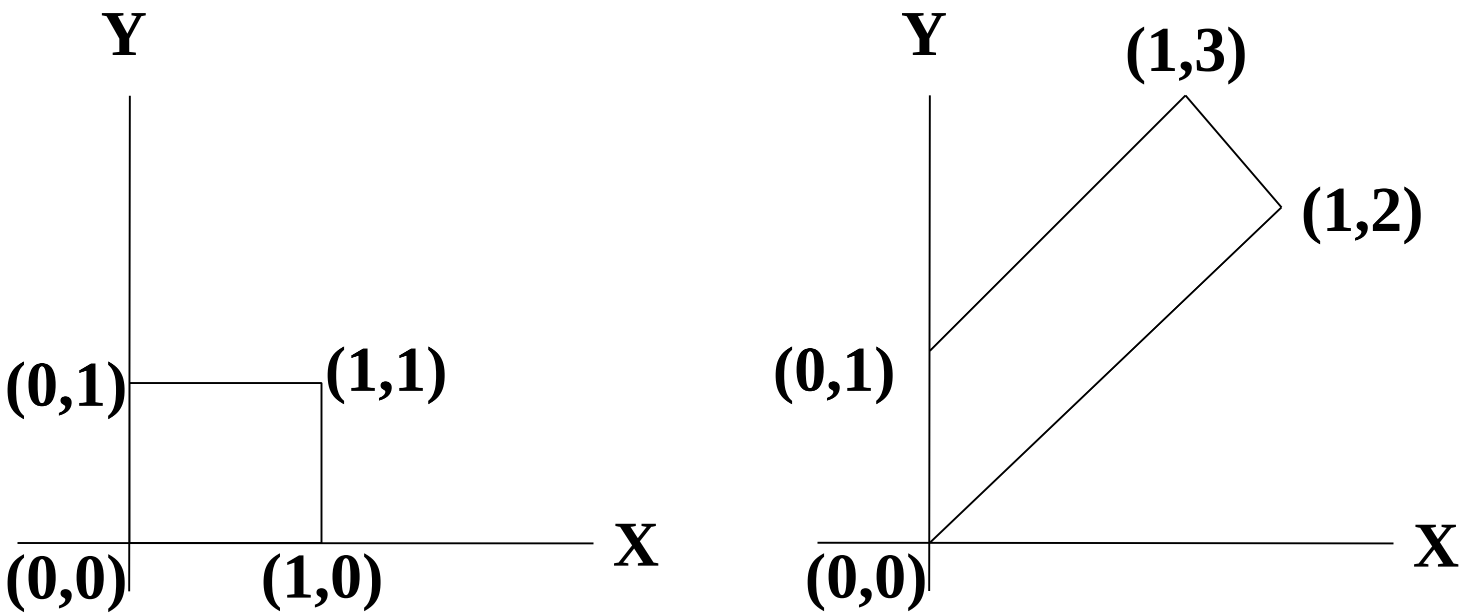
An X- direction Shear

For example, Sh_x=2



An Y- direction Shear

For example, Sh_y=2



MODULE-2

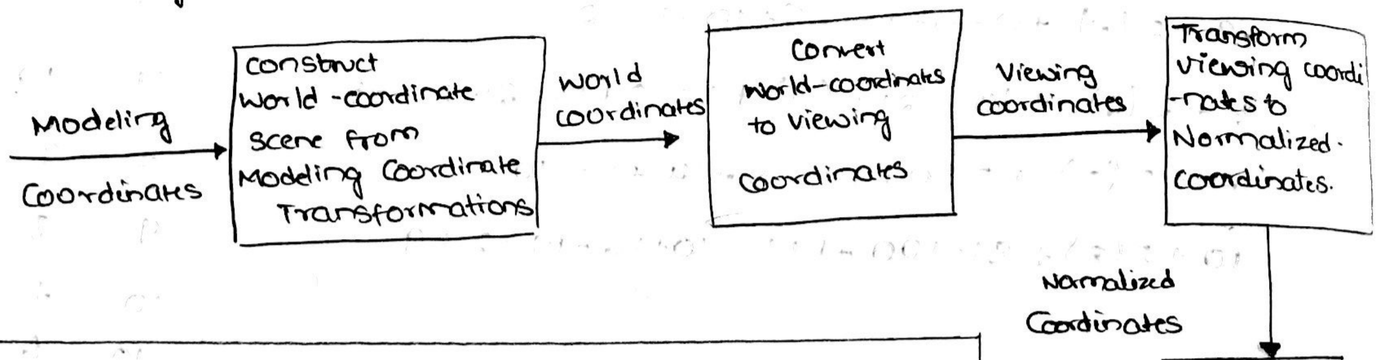
1. Polygon Filling with a color
2. 2D Transformation [Translation, Rotation, Scaling, Shearing, Reflection]
3. 2D Viewing

★ 2D viewing tranformation pipeline

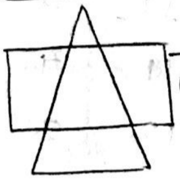
2D VIEWING

The aim is to learn how exactly we can view 2D objects and the mathematics behind conversion of world coordinates to screen co-ordinates.

2D viewing pipeline (imp)

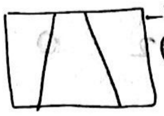


Conversion of world to screen co-ordinates



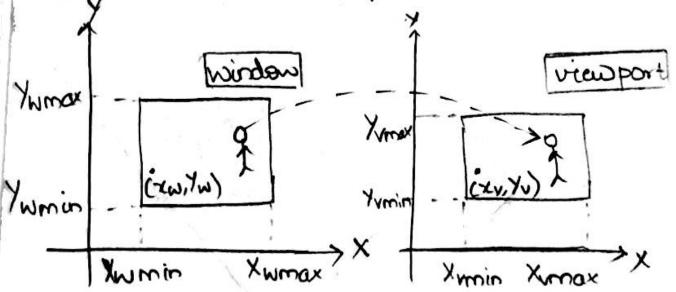
-window (what to be displayed) has → $\left\{ \begin{matrix} x_{wmin}, y_{wmin} \\ x_{wmax}, y_{wmax} \end{matrix} \right\}$ window coordinates

ViewPort



-Viewport (where to be displayed) has → $\left\{ \begin{matrix} x_{vmin}, y_{vmin} \\ x_{vmax}, y_{vmax} \end{matrix} \right\}$

So, we have to map (convert) window to viewport coordinates



Relative position will be same for both window & viewport, but size of object changes.

Since relative position is same, we can have.

for $x \rightarrow$

$$\boxed{\frac{x_w - x_{wmin}}{x_{wmax} - x_{wmin}} = \frac{x_v - x_{vmin}}{x_{vmax} - x_{vmin}}} \quad -1$$

for $y \rightarrow$

$$\boxed{\frac{y_w - y_{wmin}}{y_{wmax} - y_{wmin}} = \frac{y_v - y_{vmin}}{y_{vmax} - y_{vmin}}} \quad -2$$

we have to find corresponding viewport coordinates x_v, y_v from above equations

so from ① $\Rightarrow$ $x_v - x_{vmin} = (x_{vmax} - x_{vmin})\left(\dfrac{x_w - x_{wmin}}{x_{wmax} - x_{wmin}}\right)$

$$x_v - x_{vmin} = (x_w - x_{wmin})\left(\frac{x_{vmax} - x_{vmin}}{x_{wmax} - x_{wmin}}\right) = x_w\left(\frac{x_{vmax} - x_{vmin}}{x_{wmax} - x_{wmin}}\right) - x_{wmin}\left(\frac{x_{vmax} - x_{vmin}}{x_{wmax} - x_{wmin}}\right)$$

$$x_v - x_{vmin} = x_w\left(\frac{x_{vmax} - x_{vmin}}{x_{wmax} - x_{wmin}}\right) - \frac{x_{wmin}x_{vmax} + x_{wmin}x_{vmin}}{x_{wmax} - x_{wmin}} + x_{vmin}$$

$$x_v = x_w\left(\frac{x_{vmax} - x_{vmin}}{x_{wmax} - x_{wmin}}\right) + \frac{x_{vmin}x_{wmax} - x_{vmin}x_{wmin} + \cancel{x_{wmin}x_{vmin}} - x_{wmin}x_{vmax}}{x_{wmax} - x_{wmin}}$$

$$x_v = x_w\left(\frac{x_{vmax} - x_{vmin}}{x_{wmax} - x_{wmin}}\right) + \left(\frac{x_{wmax}x_{vmin} + x_{wmin}x_{vmax}}{x_{wmax} - x_{wmin}}\right)$$

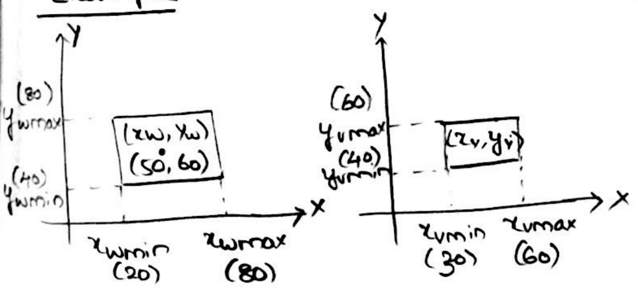
$\Rightarrow$ $\boxed{x_v = x_w S_x + T_x}$ where $S_x = \dfrac{x_{vmax} - x_{vmin}}{x_{wmax} - x_{wmin}}$

$T_x = \dfrac{x_{wmax}x_{vmin} - x_{wmin}x_{vmax}}{x_{wmax} - x_{wmin}}$

$\boxed{y_v = y_w S_y + T_y}$ where $S_y = \dfrac{y_{vmax} - y_{vmin}}{y_{wmax} - y_{wmin}}$

$T_y = \dfrac{y_{wmax}y_{vmin} - y_{wmin}y_{vmax}}{y_{wmax} - y_{wmin}}$

Example



Given: $x_{wmin} = 20$, $x_{wmax} = 80$, $y_{wmin} = 40$, $y_{wmax} = 80$

$x_{vmin} = 30$, $x_{vmax} = 60$, $y_{vmin} = 40$, $y_{vmax} = 60$

$(x_v, y_v) = ?$

① $\Rightarrow \dfrac{x_v - 30}{60 - 30} = \dfrac{50 - 20}{80 - 20} \Rightarrow x_v - 30 = 15 \Rightarrow \boxed{x_v = 45}$

② $\Rightarrow \dfrac{y_v - 40}{60 - 40} = \dfrac{60 - 40}{80 - 40} \Rightarrow y_v - 40 = 10 \Rightarrow \boxed{y_v = 50}$

Conclusion: An object which was at (x_w, y_w) (50, 60) in world coordinates, when captured by camera it got placed at screen coordinate (x_v, y_v) at (45, 5

$$x_v - x_{vmin} = (x_w - x_{wmin})\left(\frac{x_{vmax} - x_{vmin}}{x_{wmax} - x_{wmin}}\right)$$

$$\boxed{\therefore\ x_v = x_{vmin} + (x_w - x_{wmin})S_x \qquad y_v = y_{vmin} + (y_w - y_{wmin})S_y}$$

Aspect Ratio

Aspect ratio means making sure the object remains same or looks same even when the display window gets changed.

Open GL by default uses Ortho2D, WindowSize (2,2) and Window Position (0,0)

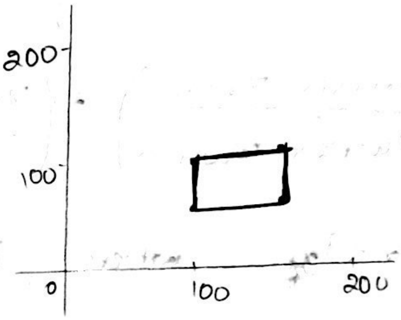
case 1:

```
#include<GL/glut.h>
void display()
{
        L  R  B  T
    gluOrtho2D(0,400,0,400);
```

500 (window) is divided into 400 units

L, B → starting position

R, T → width, height



Polygon Filling   OpenGL 2D viewing Functions

—— x ——

8-3-19

2D transformation.

translation, rotation, scaling, shearing & reflection.
[moving] [via some degree clockwise or anticlockwise] [uniform or nonuniform] [+ve or -ve]

Homogeneous coordinates

we require homogeneous coordinates to represent transformation in the form of matrix.

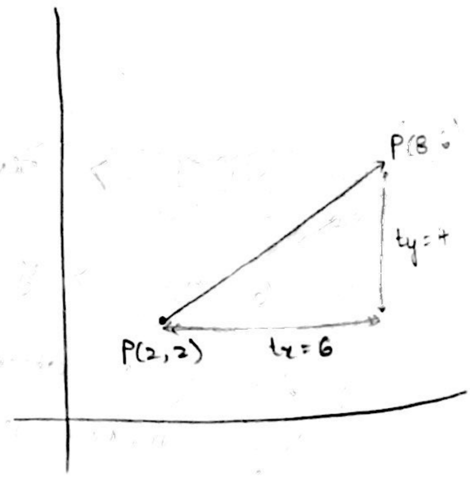
Translation

$P_x' = P_x + t_x$

$P_y' = P_y + t_y$

$P' = P + T$

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

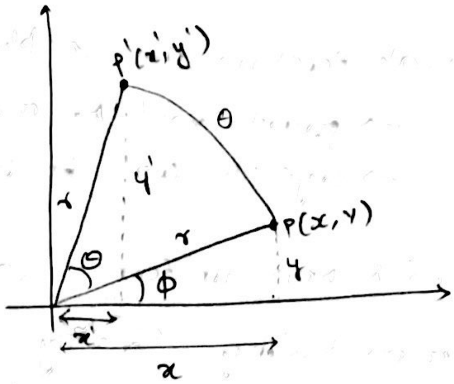


https://hemanthrajhemu.github.io

Rotation.

$\cos\phi = \frac{x}{r}$ $\sin\phi = \frac{y}{x}$

$x = r\cos\phi$ $y = r\sin\phi$



$\cos(\phi+\theta) = \frac{x'}{r}$

$x' = r\cos(\phi+\theta) = r\cos\phi\cos\theta - r\sin\phi\sin\theta$

$\boxed{x' = x\cos\theta - y\sin\theta}$

$\sin(\phi+\theta) = \frac{y'}{r}$

$y' = r\sin(\phi+\theta) = r\cos\phi\sin\theta + r\sin\phi\cos\theta = x\sin\theta + y\cos\theta$

$\boxed{y' = x\sin\theta + y\cos\theta}$

$P_x' = P_x\cos\theta - P_y\sin\theta$

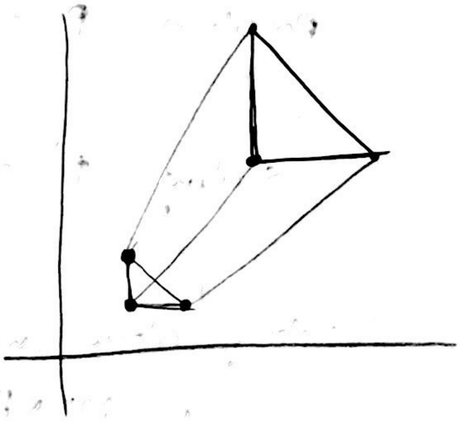
$P_y' = P_x\sin\theta + P_y\cos\theta$

$P' = R * P$

clockwise (−ve) anticlockwise (+ve)

$$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

Scaling.



$P_x' = S_x \cdot P_x$

$P_y' = S_y \cdot P_y$

In matrix form:

$P' = S * P$

scale matrix as:

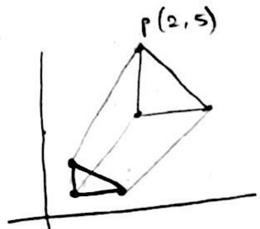
$$S = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix}$$

If scale factors are in between 0 and 1:

→ the points will be moved closer to origin

→ the object will be smaller.

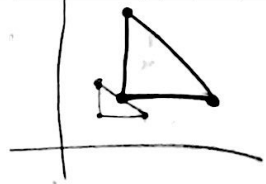
Ex: $P(2, 5)$ $S_x = 0.5$ $S_y = 0.5$



If scale factors are ~~in btw 0 & 1~~ greater than 1

→ points will be moved away from origin

→ objects will be larger



Uniform & Non uniform scaling.

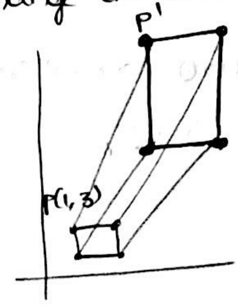
Uniform scaling $S_x = S_y$ [only change in size]

Nonuniform scaling $S_x \neq S_y$. [differential scaling]

change in size & shape

square → rectangle

$p(1,3)$ $S_x = 2$, $S_y = 5$



Summary: $p' = P + T$ [Translation]

$p' = S * P$ [Scaling]

$p' = R * P$ [Rotation]

Translation $p' = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$

Rotation $p' = \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

Scaling $p' = \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

Combing above, we can say that

$p' = M_1 * P + M_2$

Homogenous coordinates.

Using homogeneous coordinates, the transformations could be combined easily. Here we reformulate equation to eliminate matrix addition.

In homogenous coordinate system, we combine multiplicative & translational terms by expanding 2x2 matrix representation to 3x3 matrices. Also expand matrix rep for coordinate position.

8/3/19.

We represent such Cartesian coordinates (x, y) with homogeneous coordinate (x_h, y_h, h)

where $x = x_h/h$, $y = y_h/h$

$(h^* x, h^* y, h)$

Set $h = 1$

$(x, y, 1)$

Homogeneous coordinate representation for translation, scaling & rotation are as follows.

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

Translation matrix including homogeneous coordinate

(tx, ty) translation parameters along x, y

(x, y) current pos^n

(x', y') new pos^n

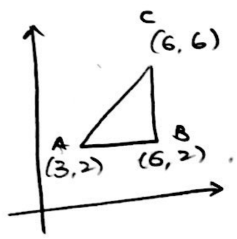
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} Sx & 0 & 0 \\ 0 & Sy & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

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

anticlock $\rightarrow \theta \rightarrow +ve$

clockwise $\rightarrow \theta \rightarrow -ve$

Q. Rotate given Δle by 90 with fixed point about origin.



Applying homogeneous coordinate system for rotation

For coordinate A(3,2)

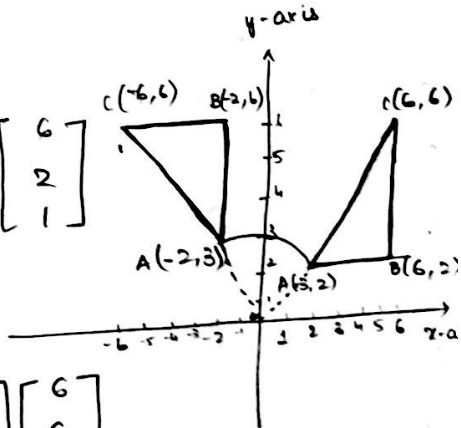
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos 90 & -\sin 90 & 0 \\ \sin 90 & \cos 90 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 3 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 3 \\ 2 \\ 1 \end{bmatrix}$$

$A(x', y', 1) = (-2, 3, 1)$

For coordinate B(6,2)

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos 90 & -\sin 90 & 0 \\ \sin 90 & \cos 90 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 6 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 6 \\ 2 \\ 1 \end{bmatrix}$$

$B(x', y', 1) = (-2, 6, 1)$



For coordinate C(6,6)

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos 90 & -\sin 90 & 0 \\ \sin 90 & \cos 90 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 6 \\ 6 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 6 \\ 6 \\ 1 \end{bmatrix}$$

$C(x', y', 1) = (-6, 6, 1)$

Prove that successive translations are additive.

If a point P is translated by T (tx_1, ty_1) to P' & then translated by (tx_2, ty_2) to P''

$$P' = T(tx_1, ty_1) * P$$

$$P'' = T(tx_2, ty_2) * P'$$

Substituting these equations we obtain

$$P'' = T(tx_2, ty_2) * (T(tx_1, ty_1) * P)$$

$$= T(tx_2, ty_2) * T(tx_1, ty_1) * P$$

Successive scaling is multiplicative

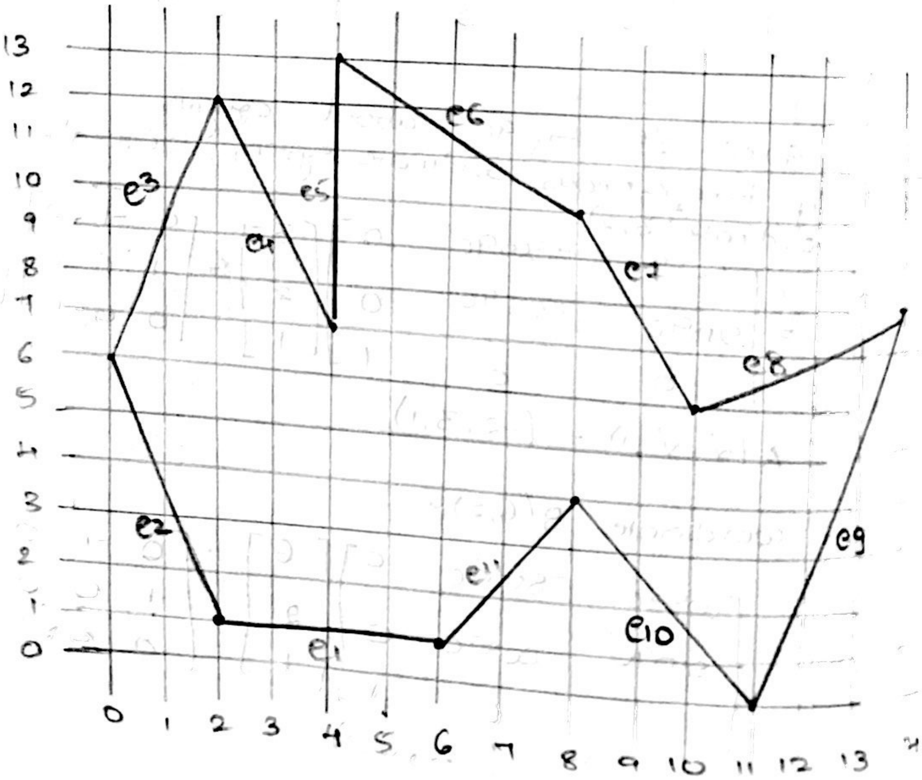
Successive ~~multiplica~~ rotation is additive

4-4-19

General Pivot point rotation : bring to origin & push it back

Polygon Data Structure.

13
12
11
10 e6
9
8
7 e4 e5
6 e3 e7 e8
5
4
3
2
1 e2 e1 e11
0 e10 e9



	xmin	ymax	1/m
e1			
e2	2	6	-2/5
e3	~~1/3~~ 0	12	1/3
e4	4	12	-2/5
e5	4	13	0
e6	6 2/3	13	-4/3
e7	10	10	-1/2
e8	10	8	2
e9	11	8	3/8
e10	11	4	-3/4
e11	6	4	2/3

e2 → (2, 1) (0, 6)

xmin → 2
ymax → 6
$1/m = \frac{-2}{5}$

e3 :- (0, 6) to (2, 12)
xmin → 0
ymax → 12
$1/m = \frac{2}{6} = \frac{1}{3}$

e4 :- (2, 12) (4, 7)
xmin → 4
ymax → 12
$1/m = -\frac{2}{5}$

Rules to be followed : 3 rules.

1.

Inside outside test : to detect whether a pixel is inside polygon or outside polygon.

odd → inside
even → outside

Non-zero winding rule

2D Composite problems:-

NOTE : If we have to rotate about origin, the eq. is
$p' = T(x, y) * R(\theta) * T(-x, -y) * P(x, y)$ ↳ for every point in polygon.

Apply above formula for all points

If we want to rotate a polygon keeping any point fixed
$p' = T(x, y) * R(\theta) * T(-x, -y) * P(x, y)$
First apply ~~fix~~ point to be fixed as $T(x, y)$ in above formula.
In final eq put other points

★ Other transformations

Reflection , shearing

Reflection : It is producing a mirror object

case 1

Reflection about x-axis $\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

case : 2

y-axis $\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

case : 3

Reflect (about) of object relative to an axis $\perp^r$ to XY plane & passing through coordinate origin.

$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Module 2

- ✓ Polygon filling
- ✓ 2D Transformation
- 2D Viewing

Module 3

- Illumination
- 3D transformations
- ✓ Clipping - 5 ways
  - ↳ point, line, polygon, curve, text

Module 3

Clipping

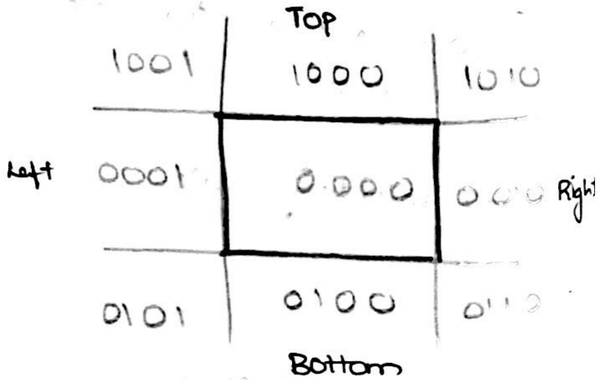
→ Point → Line → Polygon → curve → text

Line — Cohen-Sutherland algo

Polygon — Sutherland Hodgeman

Cohen-Sutherland algo

T B R L



Test using bitwise functions.

if $C_0 | C_1 = 0000$ accept (draw)

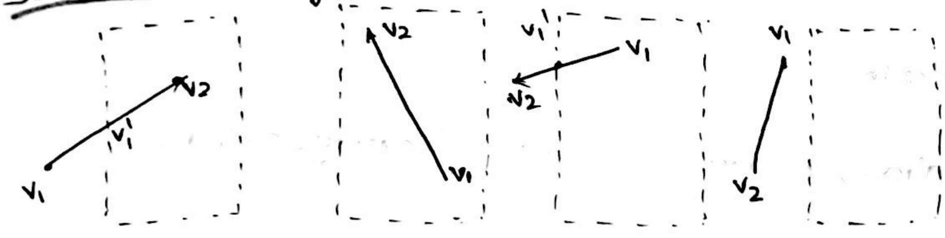
else if $C_0 \& C_1 \neq 0000$

reject (don't draw)

else clip & retest

Challenge 1 : To find intersection points

Sutherland Hodgeman

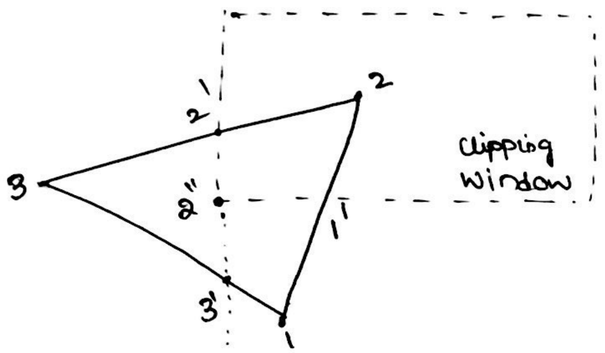


out → in — Output : V_1', V_2

in → in — Output : V_2

in → out — Output : V_1'

out → out — Output : none.



[1,2] : (in - in) → [2]

[2,3] : (in - out) → [2']

[3,1] : (out - in) → [3', 1]

[2,2'] : (in-in) → [2']

[2',3'] : (in-in) → [3']

[3',1] : (in-in) → [1]

[1,2] : (in-in) → [2]

[2',3'] : (in-out) → [2'']

[3',1] : (out-out) → [ ]

[1,2] : (out-in) → [1',2]

[2,2']

[2'', 1'] in → in → 1'