2D TRANSFORMATIONS AND MATRICES

Representation of Points:

2 x 1 matrix: $\begin{vmatrix} x \\ y \end{vmatrix}$

General Problem: |B| = |T| |A|

|T| represents a generic operator to be applied to the points in A. T is the geometric transformation matrix. A & T are know, want to find B, the transformed points.

General Transformation of 2D points:

$$\begin{bmatrix} a & c \\ b & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x' \\ y' \end{bmatrix}$$

$$x' = ax + cy$$

$$y' = bx + dy$$

Special cases of 2D Transformations:

1) T= identity matrix, a=d=1, b=c=0

$x'=x$, $y'=y$

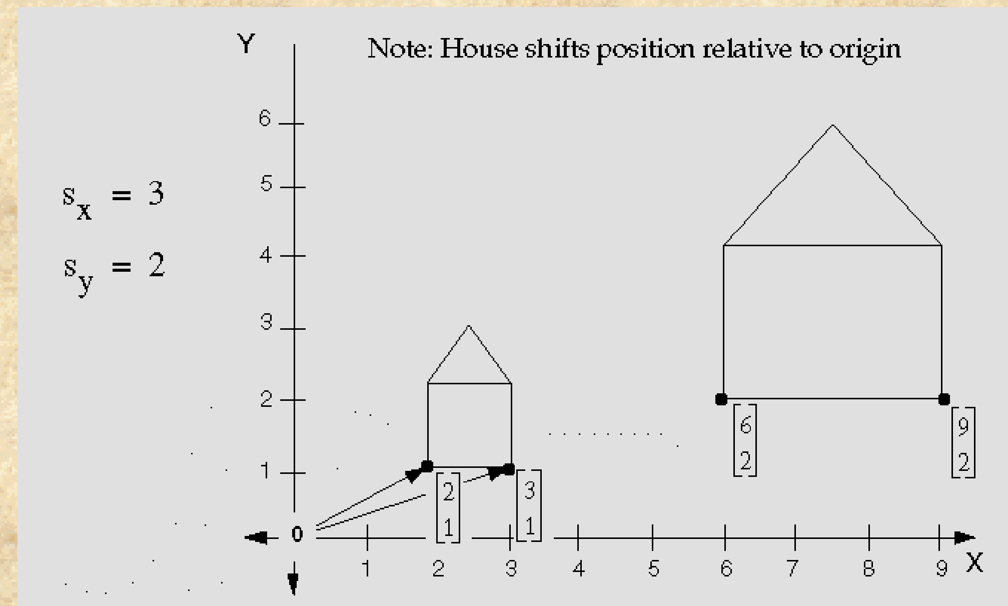
so far, what we would expect!

2) *Scaling & Reflections*: b=0, c=0

$x' = a.x$, $y' = d.y$; This is scaling by a in x, d in y.

Scale matrix: let S_x =a, S_y=d

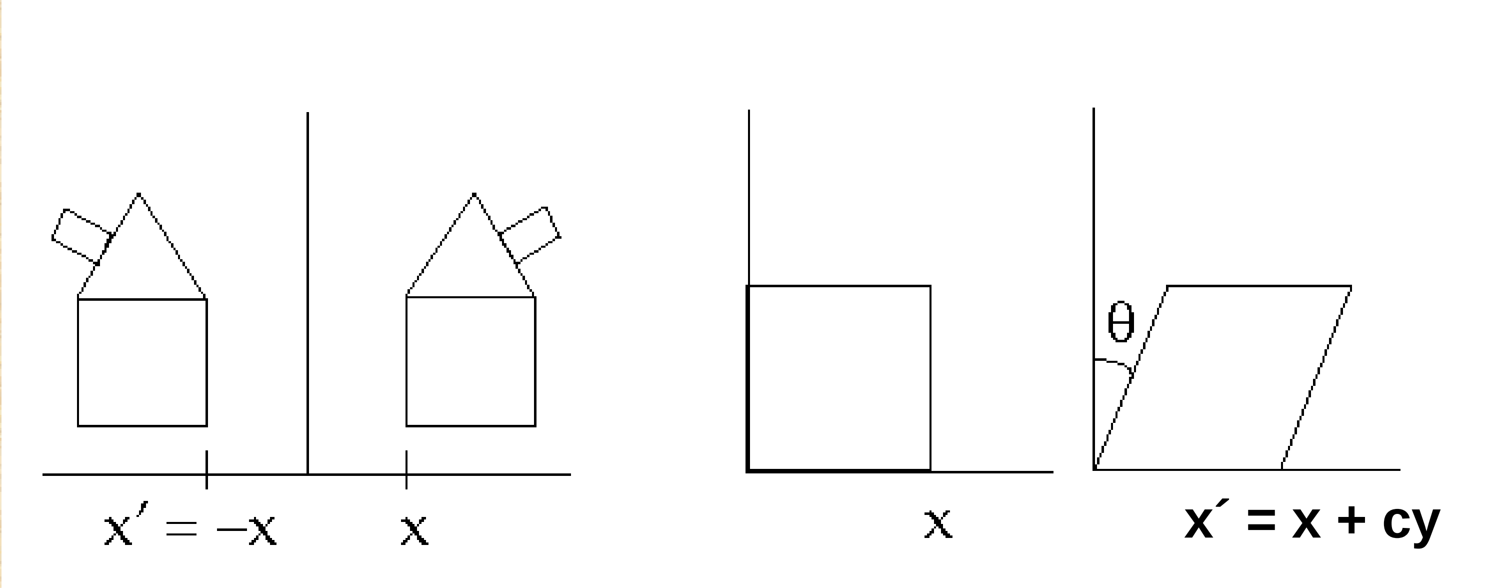
$$\begin{vmatrix} S_x & 0 \\ 0 & S_y \end{vmatrix}$$



What if S_x and/ or $S_y < 0$?

Get reflections through an axis or plane

Only diagonal terms involved in scaling and reflections



Off diagonal terms: <u>Shearing</u>

a = d = 1

let, c = 0, b = 2

x' = x
y' = bx + y

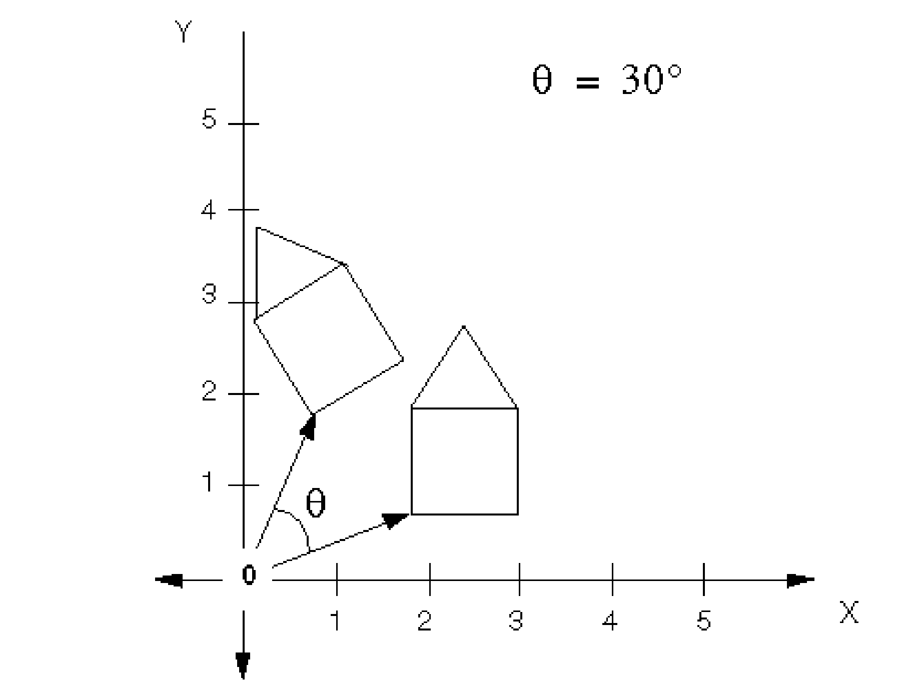
y' depends linearly on x

Similarly for b=0, c not equal to zero.

$$\begin{bmatrix} a & c \\ b & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x' \\ y' \end{bmatrix}$$

$$x' = ax + cy$$

$$y' = bx + dy$$



ROTATION

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

In matrix form, this is :

$$\begin{vmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{vmatrix}$$

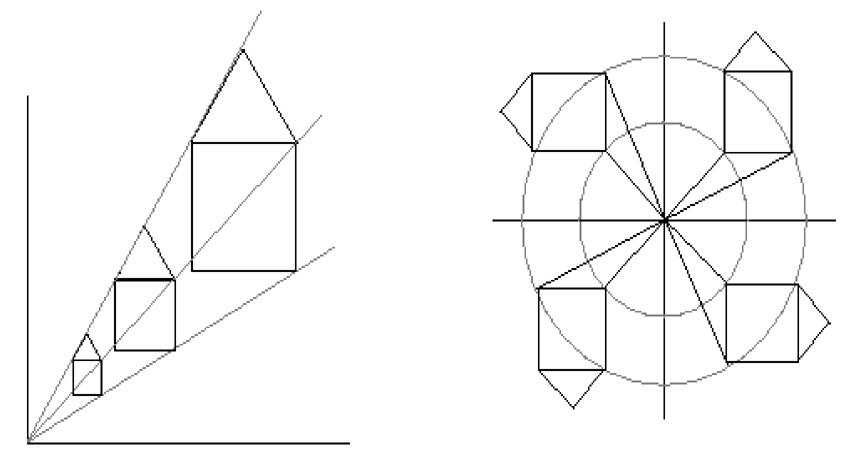
Positive Rotations: counter clockwise about the origin

For rotations, $\det|T| = 1$ and $|T|^T = |T|^{-1}$

Translations

$B = A + T_r$, where $T_r = |tx\ ty|^T$

Note: we can not directly represent translations as matrix multiplication, as we can rotations and scalings



Where else are translations introduced?

1) **Rotations - when object not centered at the origin.**
2) **Scaling - when objects / lines not centered at the origin.**
   - **line from (2,1) to (4,1) scaled by 2 in x & y.**
   - **If line intersects the origin, no translation.**
   - **Scaling is about the origin.**

Can we represent translations in our general transformation matrix?

Yes, by using homogeneous coordinates

HOMOGENEOUS COORDINATES

We have x' = ax + cy + tx
y' = bx + cy + ty
Use a 3 x 3 matrix:

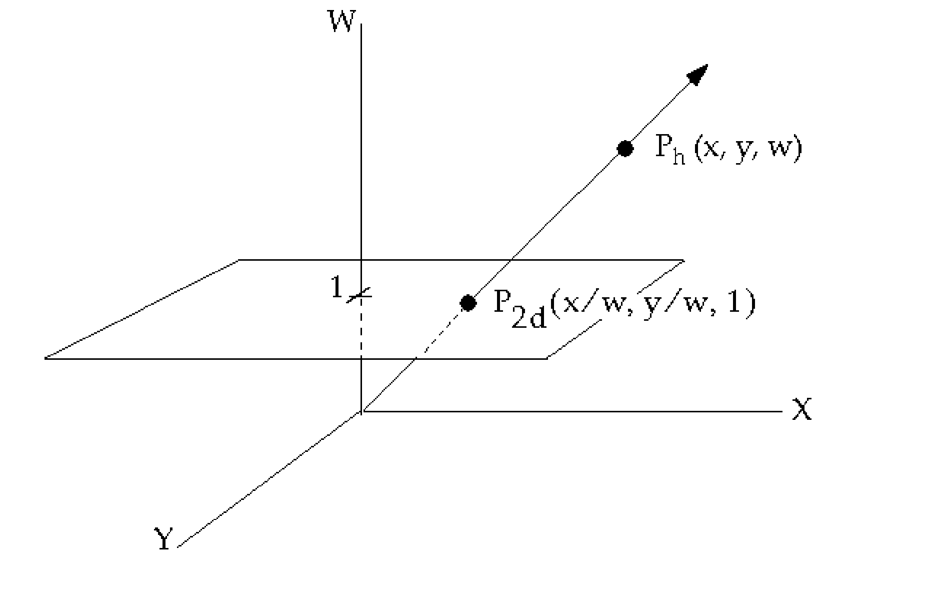
$$\begin{bmatrix} x' \\ y' \\ Z' \end{bmatrix} = \begin{bmatrix} a & c & t_x \\ b & d & t_y \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Each point is now represented by a triple: (x, y, W)
x/W, y/W are called the Cartesian coordinates of the homogeneous points.
Two homogeneous coordinates (x1, y1, w1) & (x2,2y, w2) may represent the same point, iff they are multiples of one another: (1,2,3) & (3,6,9).
There is no unique homogeneous representation of a point.

All triples of the form (tx, ty, tW) form a line in x,y,W space.

Cartesian coordinates are just the plane w=1 in this space.

W=0, are the points at infinity



COMPOSITE TRANSFORMATIONS

If we want to apply a series of transformations T1, T2, T3 to a set of points, We can do it 2 ways:

1) We can calculate p'=T1*p, p'' = T2*p', p'''=T3*p''

2) Calculate T= T1*T2*T3, then p'''= T*p.

Method 2, saves large number of adds and multiplies. Approximately 1/3 as many operations. Therefore, we concatenate or compose the matrices into one final transformation matrix that we apply to the points.

Translations:

Translate the points by tx1, ty1, then by tx2, ty2:

$$\begin{bmatrix} 1 & 0 & (tx1 + tx2) \\ 0 & 1 & (tx1 + tx2) \\ 0 & 0 & 1 \end{bmatrix}$$

Scaling: Similar to translations

Rotations:

Rotate by q1, then by q2, stick the (q1+q2) in for q, or calculate T1 for q1, then T2 for q2 & multiply them. Gives same result - work it out (exercise).

Rotation about an arbitrary point P in space

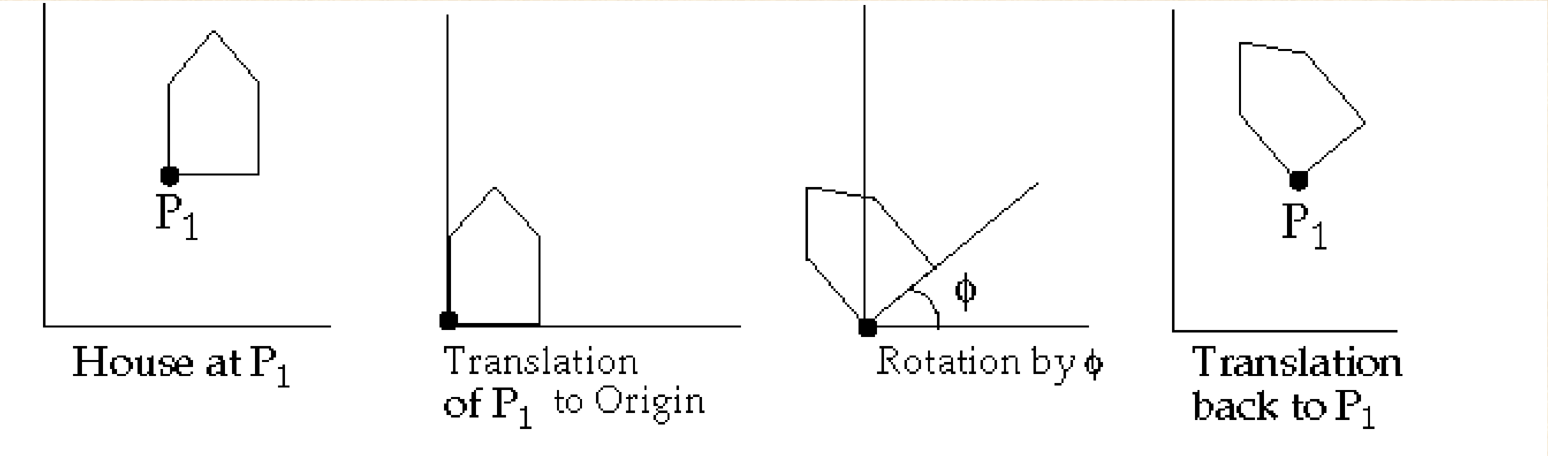
As we mentioned before, rotations are about the origin. So to rotate about a point P in space, translate so that P coincides with the origin, then rotate, then translate back:

- **Translate by (-Px, -Py)**
- **Rotate**
- **Translate by (Px, Py)**

T = T1(Px,Py) * T2(q) * T3(-Px, -Py)

$$= \begin{bmatrix} 1 & 0 & Px \\ 0 & 1 & Py \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & -Px \\ 0 & 1 & -Py \\ 0 & 0 & 1 \end{bmatrix}$$

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



Scaling about an arbitrary point in Space

Again,

- **Translate P to the origin**
- **Scale**
- **Translate P back**

T = T1(Px,Py)* T2(sx, sy)*T3(-Px, -Py)

T =

$$\begin{bmatrix} Sx & 0 & \{Px*(1-Sx)\} \\ 0 & Sy & \{Py*(1-Sy)\} \\ 0 & 0 & 1 \end{bmatrix}$$

Commutivity of Transformations

If we scale, then translate to the origin, then translate back, is that equivalent to translate to origin, scale, translate back?

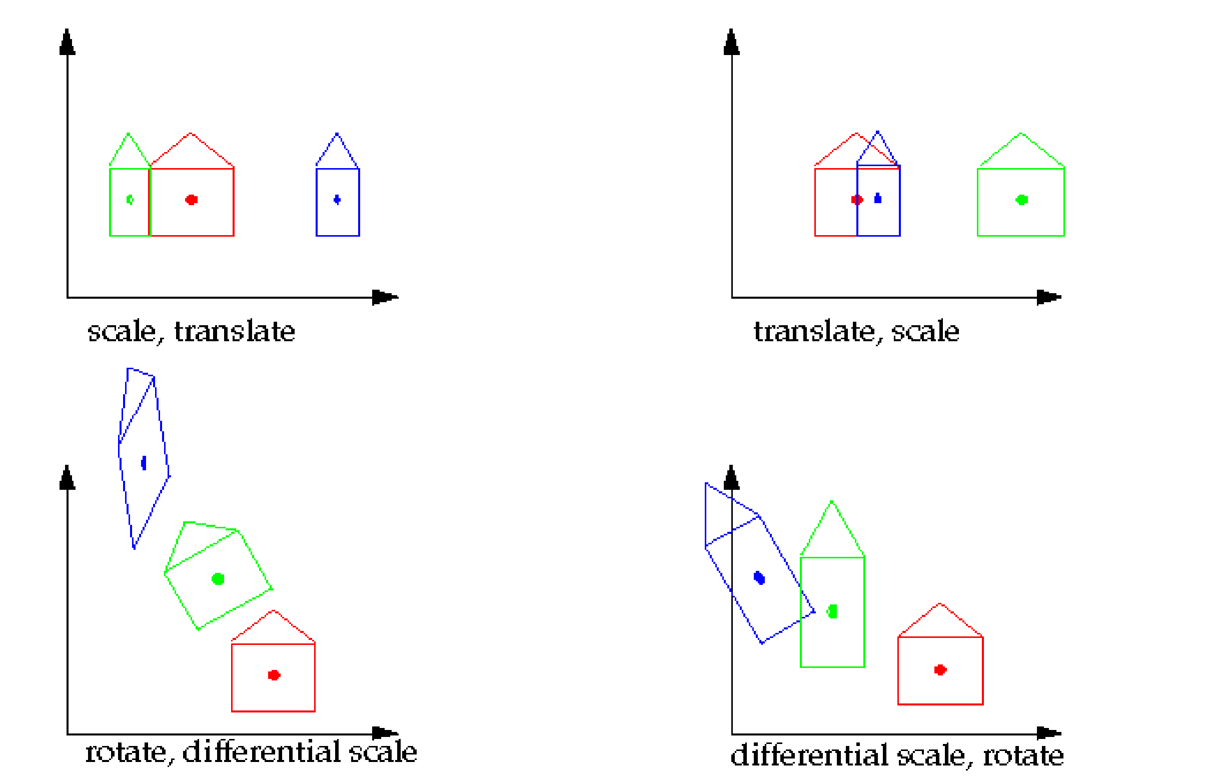
When is the order of matrix multiplication unimportant?

When does T1*T2 = T2*T1?

Cases where T1*T2 = T2*T1:

Order: R-G-B

T1	T2
translation	translation
scale	scale
rotation	rotation
scale(uniform)	rotation



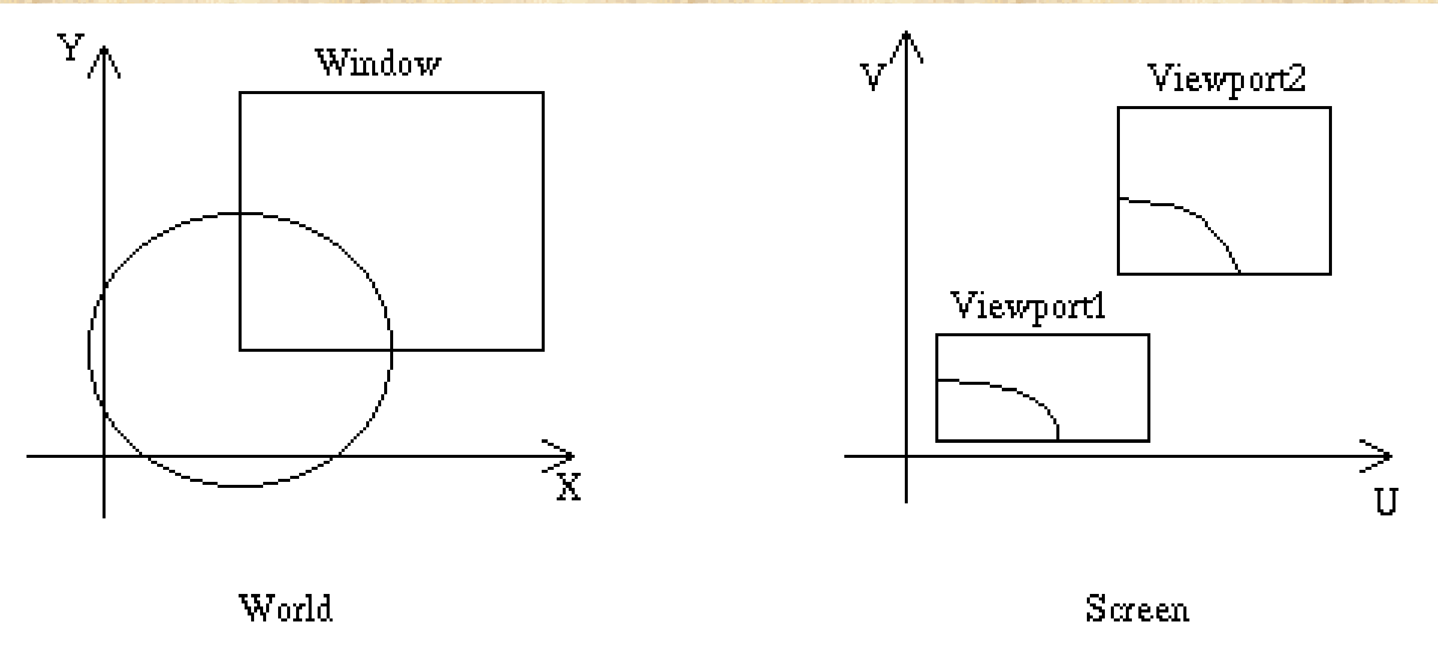
COORDINATE SYSTEMS

Screen Coordinates: The coordinate system used to address the screen (device coordinates)

World Coordinates: A user-defined application specific coordinate system having its own units of measure, axis, origin, etc.

Window: The rectangular region of the world that is visible.

Viewport: The rectangular region of the screen space that is used to display the window.



WINDOW TO VIEWPORT TRANSFORMATION

Want to find the transformation matrix that maps the window in world coordinates to the viewport in screen coordinates.

Viewport: (u, v space) denoted by: $u_{min}, v_{min}, u_{max}, v_{max}$

Window: (x, y space) denoted by: $x_{min}, y_{min}, x_{max}, y_{max}$

$$M_{WV} = T(U_{\min}, V_{\min}) * S(S_x, S_y) * T(-x_{\min}, -y_{\min});$$

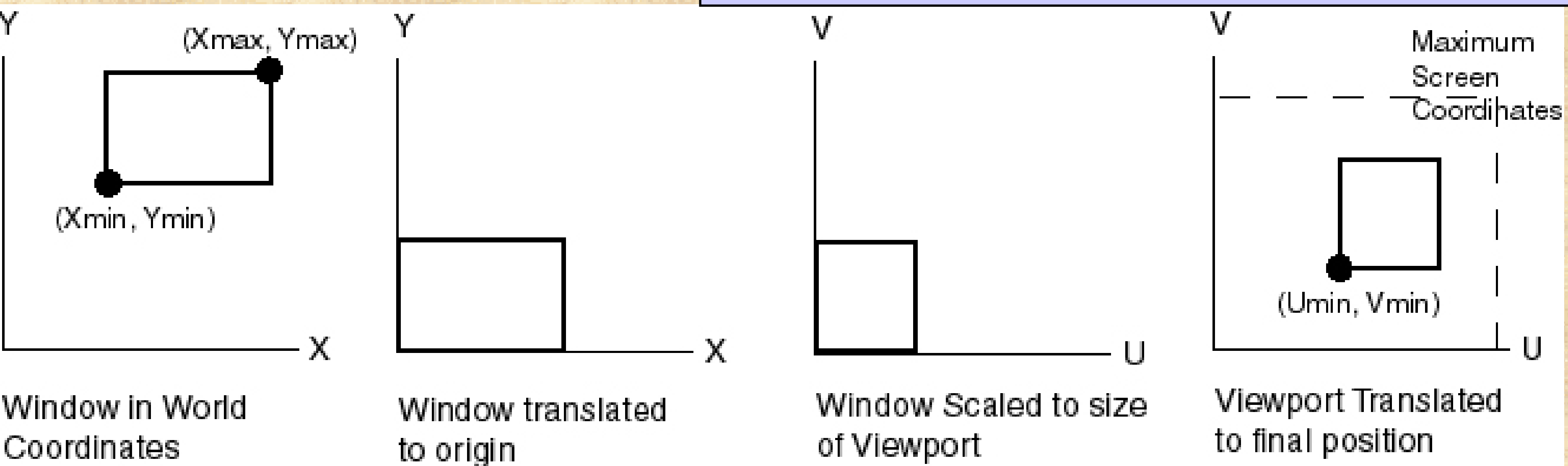
$$S_x = (U_{\max} - U_{\min}) / (x_{\max} - x_{\min});$$

$$S_y = (V_{\max} - V_{\min}) / (y_{\max} - y_{\min});$$

$$M_{WV} = \begin{bmatrix} S_x & 0 & (-x_{\min} * S_x + U_{\min}) \\ 0 & S_y & (-y_{\min} * S_y + V_{\min}) \\ 0 & 0 & 1 \end{bmatrix}$$

The transformation:

- **Translate the window to the origin**
- **Scale it to the size of the viewport**
- **Translate it to the viewport location**



Transformations of Parallel Lines