

UNIT 2: 2D Transformations, Viewing & Clipping

Introduction of Transformations

- Computer Graphics provide the facility of viewing object from different angles. The architect can study building from different angles i.e.
 - Front Evaluation
 - Side elevation
 - Top plan
- A Cartographer can change the size of charts and topographical maps. So if graphics images are coded as numbers, the numbers can be stored in memory.
- These numbers are modified by mathematical operations called as Transformation.

- The purpose of using computers for drawing is to provide facility to user to view the object from different angles, enlarging or reducing the scale or shape of object called as Transformation.
- Two essential aspects of transformation are given below:
 - Each transformation is a single entity. It can be denoted by a unique name or symbol.
 - It is possible to combine two transformations, after connecting a single transformation is obtained, e.g., A is a transformation for translation. The B transformation performs scaling. The combination of two is $C=AB$. So C is obtained by concatenation property.

- There are two complementary points of view for describing object transformation.
 - **Geometric Transformation:** The object itself is transformed relative to the coordinate system or background. The mathematical statement of this viewpoint is defined by geometric transformations applied to each point of the object.
 - **Coordinate Transformation:** The object is held stationary while the coordinate system is transformed relative to the object. This effect is attained through the application of coordinate transformations.

- Types of Transformations:
 - Translation
 - Scaling
 - Rotating
 - Reflection
 - Shearing

1. Translation

- We perform a translation on a single coordinate point by adding offsets to its coordinates so as to generate a new coordinate position.
- It is the straight line movement of an object from one position to another is called Translation. Here the object is positioned from one coordinate location to another.

- To translate a 2D position, we add translation distances t_x and t_y (that pair is also called translation vector or shift vector) to the original coordinates (x,y) to obtain the new coordinate position (x',y') .

$$x' = x + t_x, \quad y' = y + t_y$$

$$\mathbf{P} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \quad \mathbf{P}' = \begin{bmatrix} x'_1 \\ x'_2 \end{bmatrix}, \quad \mathbf{T} = \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

$$\mathbf{P}' = \mathbf{P} + \mathbf{T}$$

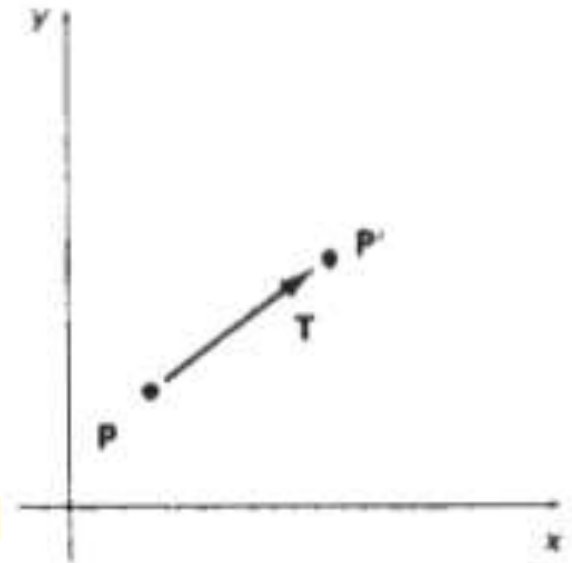


Figure 5-1
Translating a point from position \mathbf{P} to position \mathbf{P}' with translation vector \mathbf{T} .

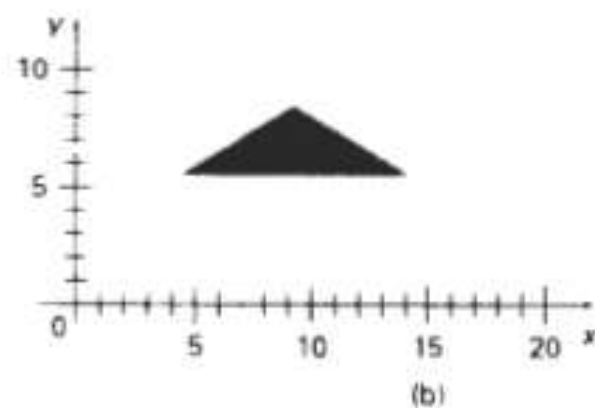
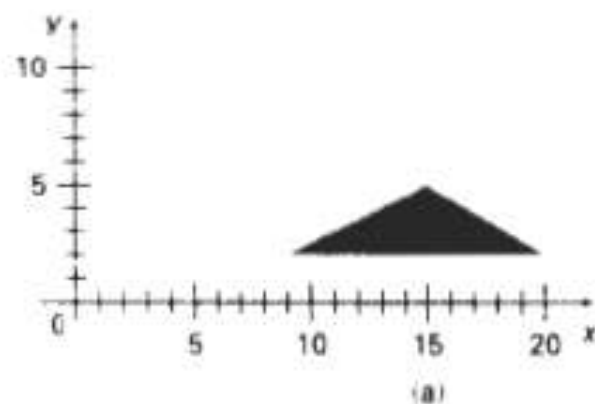


Figure 5-2
Moving a polygon from position (a)
to position (b) with the translation
vector $(-5.50, 3.75)$.

2. Scaling

- To alter the size of an object, we apply a scaling transformation.
- A simple two-dimensional scaling operation is performed by multiplying object position (x,y) by scaling factors s_x and s_y to produce the transformed coordinates (x',y') .

$$x' = x \cdot s_x, \quad y' = y \cdot s_y$$

- Scaling factor s_x scales an object in the x direction, while s_y scales in the y direction.
- The basic two-dimensional scaling equation can also be written in the following matrix form:

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

$$P' = S \cdot P$$

- Any positive values can be assigned to the scaling factors s_x and s_y .
- Values less than 1 reduce the size of objects; values greater than 1 produce enlargements.
- Specifying a value of 1 for both s_x and s_y leaves the size of objects unchanged.

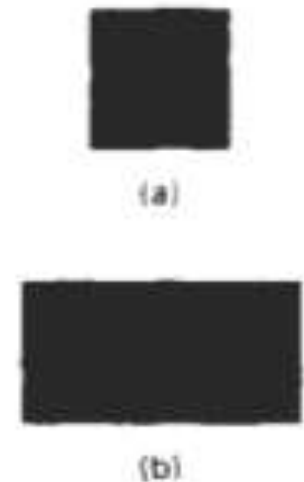


Figure 5-6
Turning a square (a) into a rectangle (b) with scaling factors $s_x = 2$ and $s_y = 1$.

- When s_x and s_y are assigned the same value, a uniform scaling is produced, which maintains relative object proportions.
- Unequal values for s_x and s_y result in a differential scaling that is often used in design applications, where pictures are constructed from a few basic shapes that can be adjusted by scaling and positioning transformation.

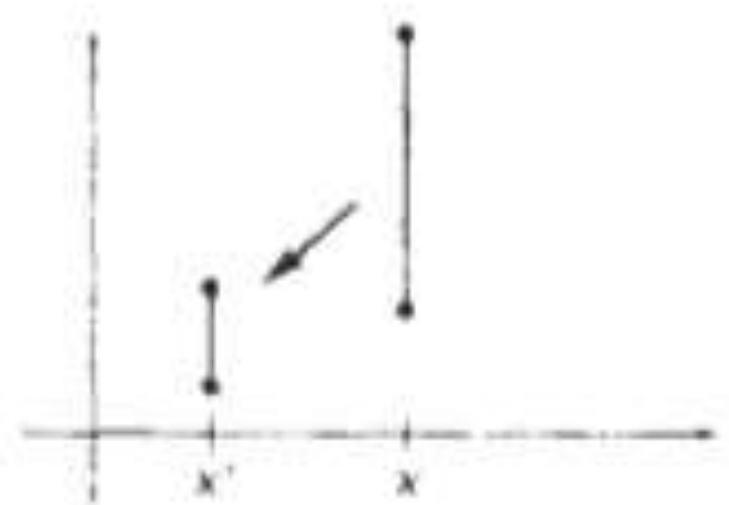
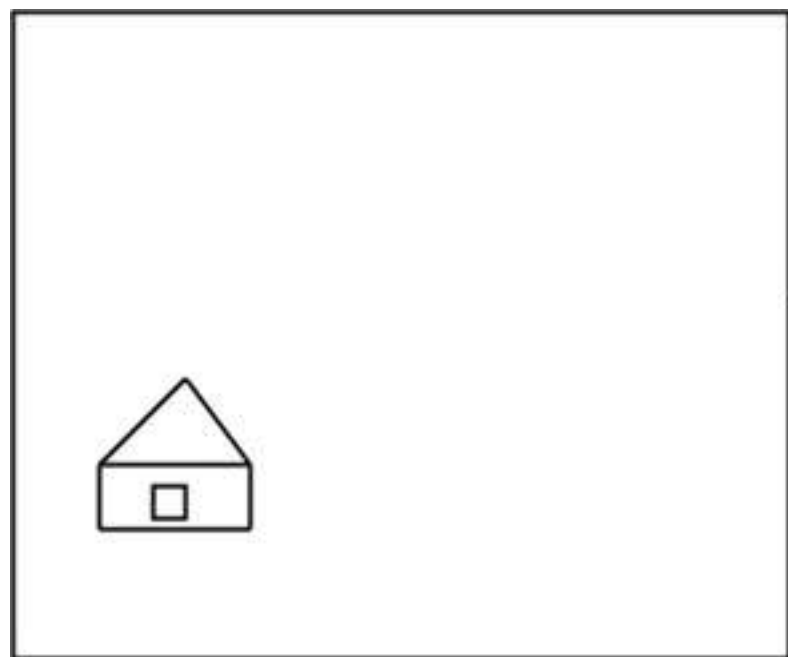
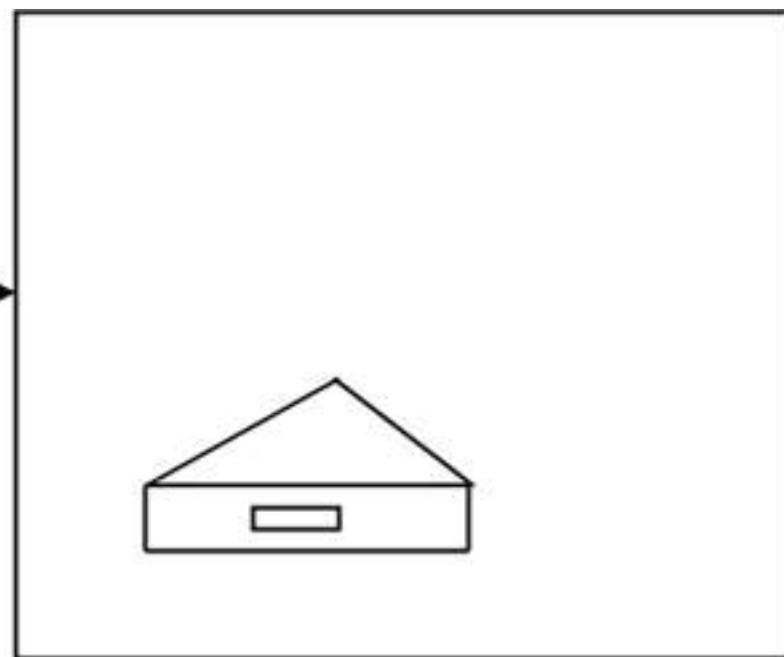


Figure 5-7
A line scaled with Eq 5-12 using $s_x = s_y = 0.5$ is reduced in size and moved closer to the coordinate origin.



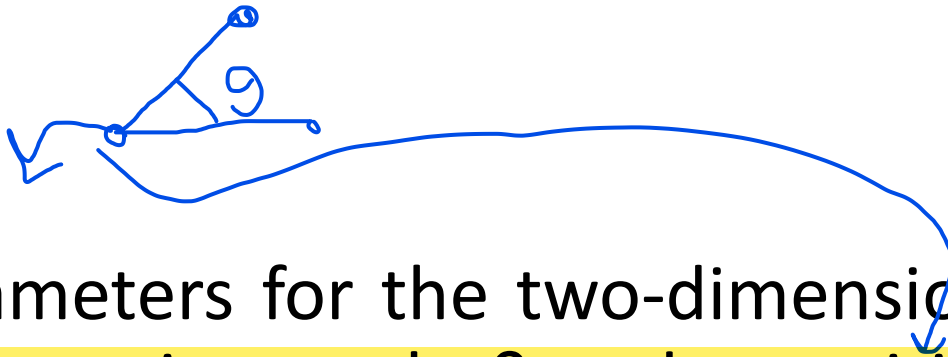
Original Object



Object after scaling in X direction

3. Rotation

- We generate a rotation transformation of an object by specifying a rotation axis and a rotation angle.
- All the points of the object are then transformed to new positions by rotating the points through the specified angle about the rotation axis.
- A two-dimensional rotation of an object is obtained by repositioning the object along a circular path in the xy plane.



- Parameters for the two-dimensional rotation are the rotation angle θ and a position (x_r, y_r) , called the rotation point or pivot point, about which the object is to be rotated .
- The pivot point is the intersection position of the rotation axis with the xy plane.
- A positive value for the angle θ defines a counterclockwise rotation about the pivot point and negative value rotates objects in the clockwise direction.

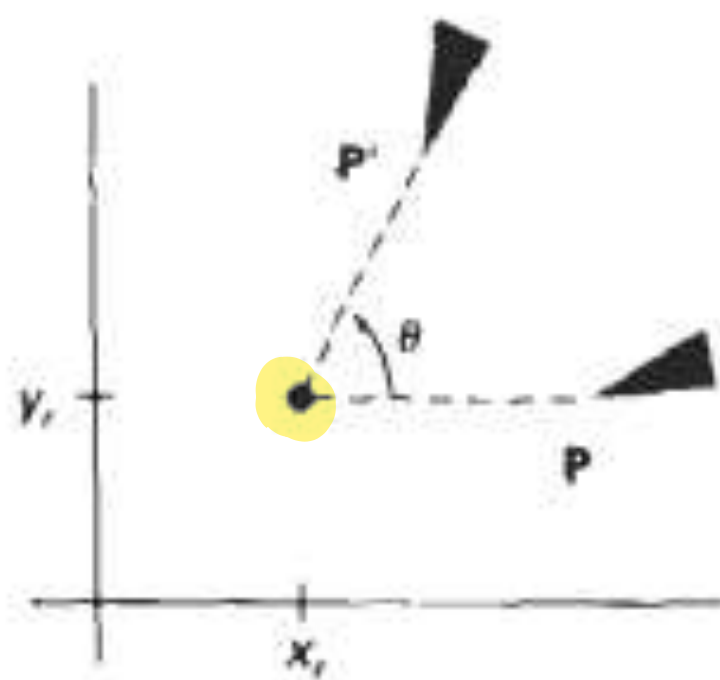


Figure 5-3
Rotation of an object through
angle θ about the pivot point
 (x_r, y_r) .

- In the figure, r is the constant distance of the point from the origin, angle ϕ is the original angle position of the point from the horizontal, and θ is the rotation angle.
- Using standard trigonometric identities we can express as:

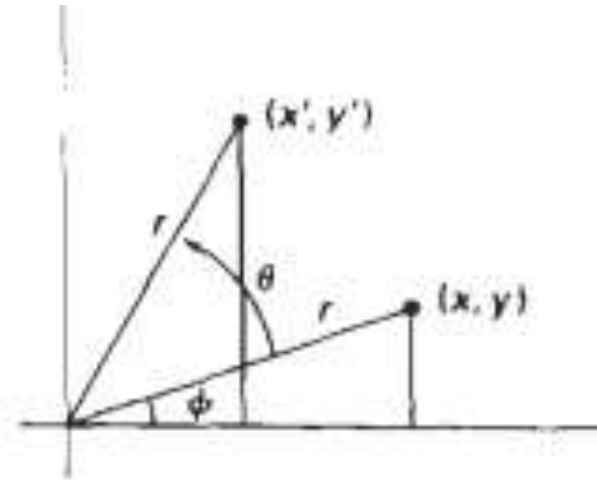


Figure 5-4
Rotation of a point from position (x, y) to position (x', y') through an angle θ relative to the coordinate origin. The original angular displacement of the point from the x axis is ϕ .

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

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

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

$$x' = x \cos \theta - y \sin \theta$$

$$y' = x \sin \theta + y \cos \theta$$

$$\mathbf{P}' = \mathbf{R} \cdot \mathbf{P}$$

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

Homogeneous Coordinates System

- Many graphics applications involve sequences of geometric transformations.
- In design and picture construction applications, we perform translations, rotations, and scalings to fit the picture components into their proper positions.

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

- We can combine the multiplicative and translational terms for 2D geometric transformations into a single matrix representation by expanding the 2x2 matrix representations to 3x3 matrices.
- To express any 2D transformation as a matrix multiplication, we represent each Cartesian coordinate position (x, y) with the homogeneous coordinate triple (x_h, y_h, h) , where:

$$x = \frac{x_h}{h} \quad y = \frac{y_h}{h}$$

- For 2D geometric transformations, we can choose the homogeneous parameter h to be any nonzero value.
- Thus, there is an infinite number of equivalent homogeneous representations for each coordinate point (x, y) .
- A convenient choice is simple to set $h = 1$. Each 2D position is then represented with homogeneous coordinates $(x, y, 1)$.
- The term homogeneous coordinates is used in mathematics to refer to the effect of this representation in Cartesian equations.

IMP

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

$$\mathbf{P}' = \mathbf{T}(t_x, t_y) \cdot \mathbf{P}$$

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

$$\mathbf{P}' = \mathbf{R}(\theta) \cdot \mathbf{P}$$

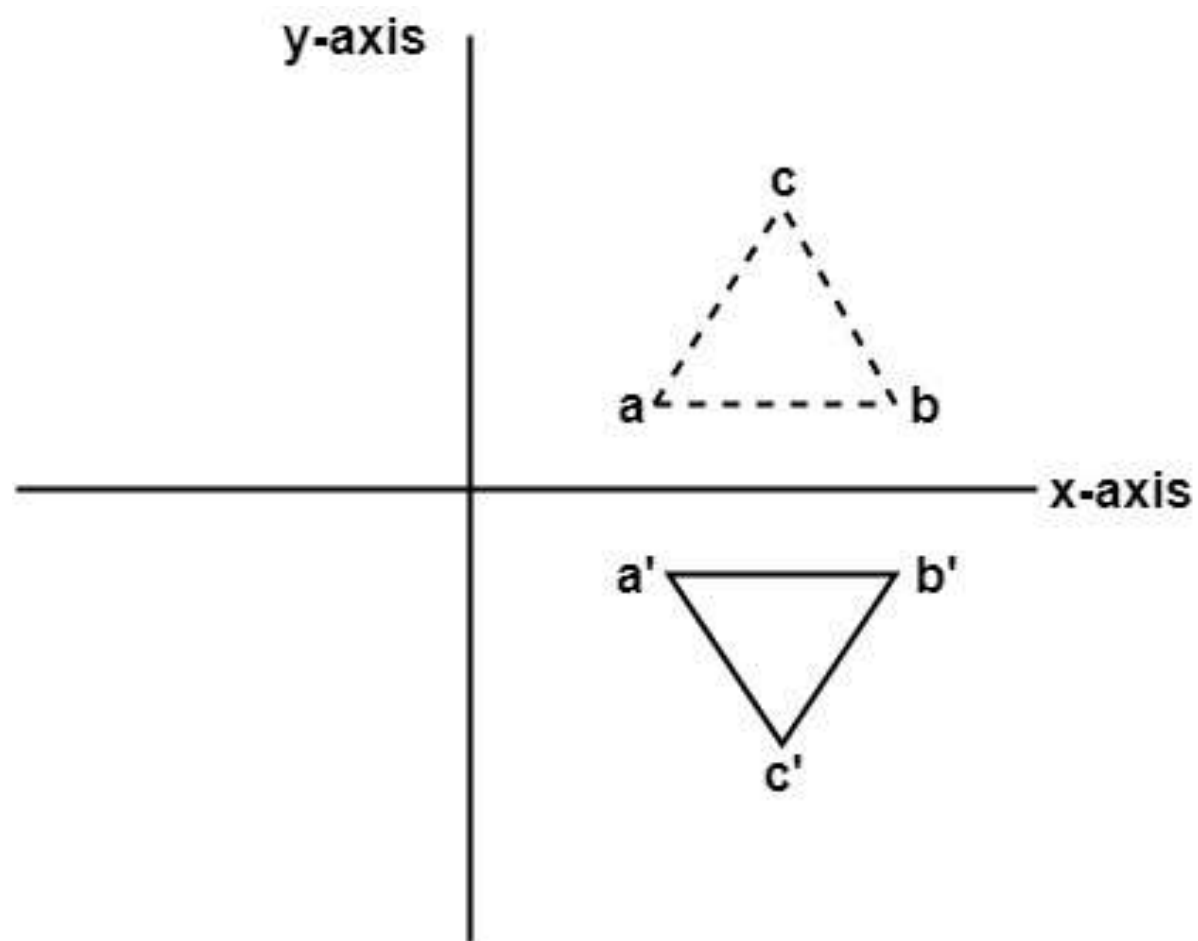
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

$$\mathbf{P}' = \mathbf{S}(s_x, s_y) \cdot \mathbf{P}$$

Other transformations: Reflection

- It is a transformation which produces a mirror image of an object. The mirror image can be either about x-axis or y-axis. The object is rotated by 180° .
- When the reflection axis is a line in the xy plane, the rotation path about this axis is in a plane perpendicular to the xy plane.
- For reflection axes that are perpendicular to the xy plane, the rotation path is in the xy plane.





- Reflection about the line $y = 0$, the x axis, is accomplished with the transformation matrix:

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

- Reflection about the y axis flips x coordinates while keeping y coordinates the same. The matrix for this transformation is:

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

4. Reflection...

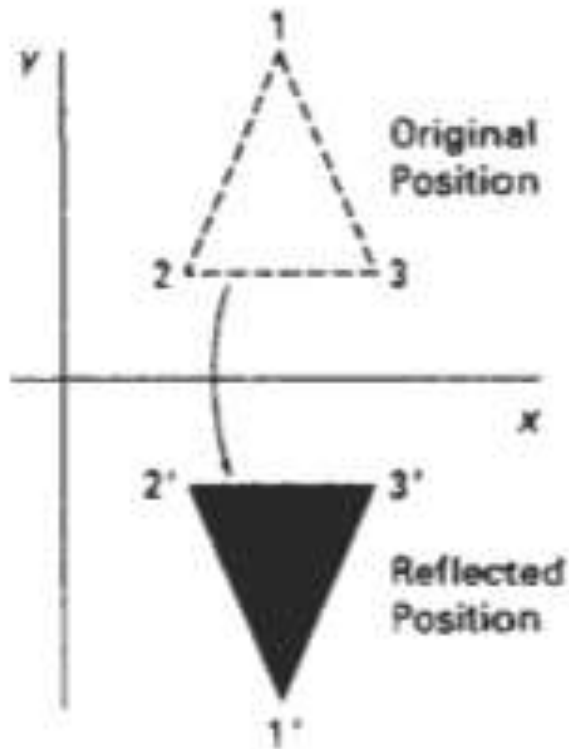


Figure 5-16
Reflection of an object about
the x axis.

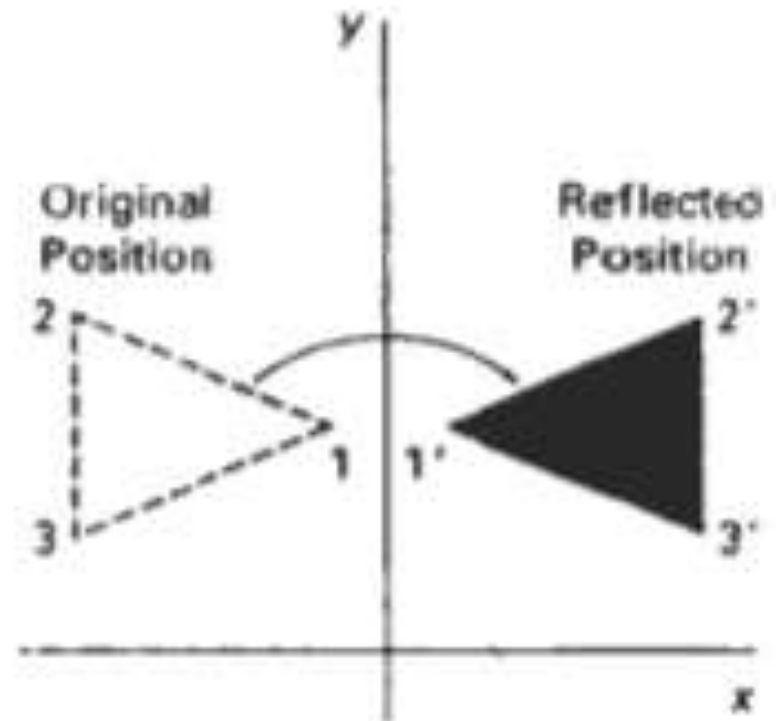


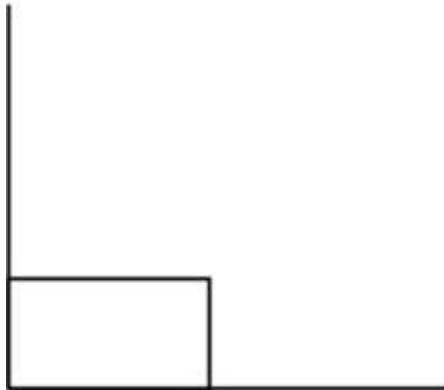
Figure 5-17
Reflection of an object about
the y axis.

5. Shearing

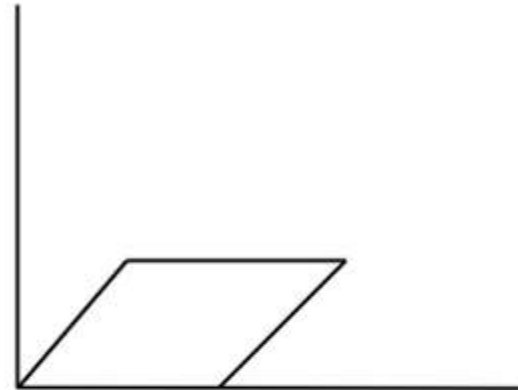
- A transformation that distorts the shape of an object such that the transformed shape appears as if the object were composed in the internal layers that had been caused to slide over each other is called a shear.
- Two common shearing transformations are those that shift coordinate x values and those that shift y values.

5. Shearing...

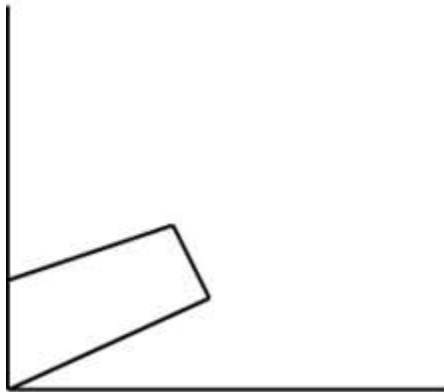
Important



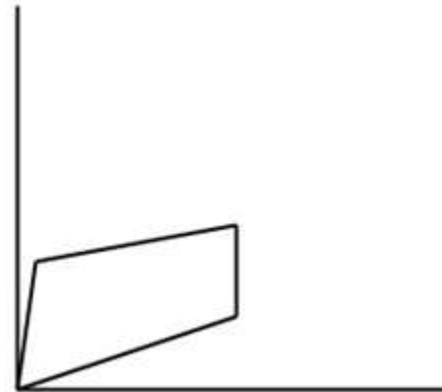
Original Object



Shear in X direction



Shear in Y direction



Shear in both directions

- An x-direction shear relative to the x axis is produced with the transformation matrix:

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

- Which transforms coordinate position as:

$$x' = x + sh_x \cdot y, \quad y' = y$$

- We can generate x-direction shears relative to other reference lines with:

$$\begin{bmatrix} 1 & sh_x & -sh_x \cdot y_{ref} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

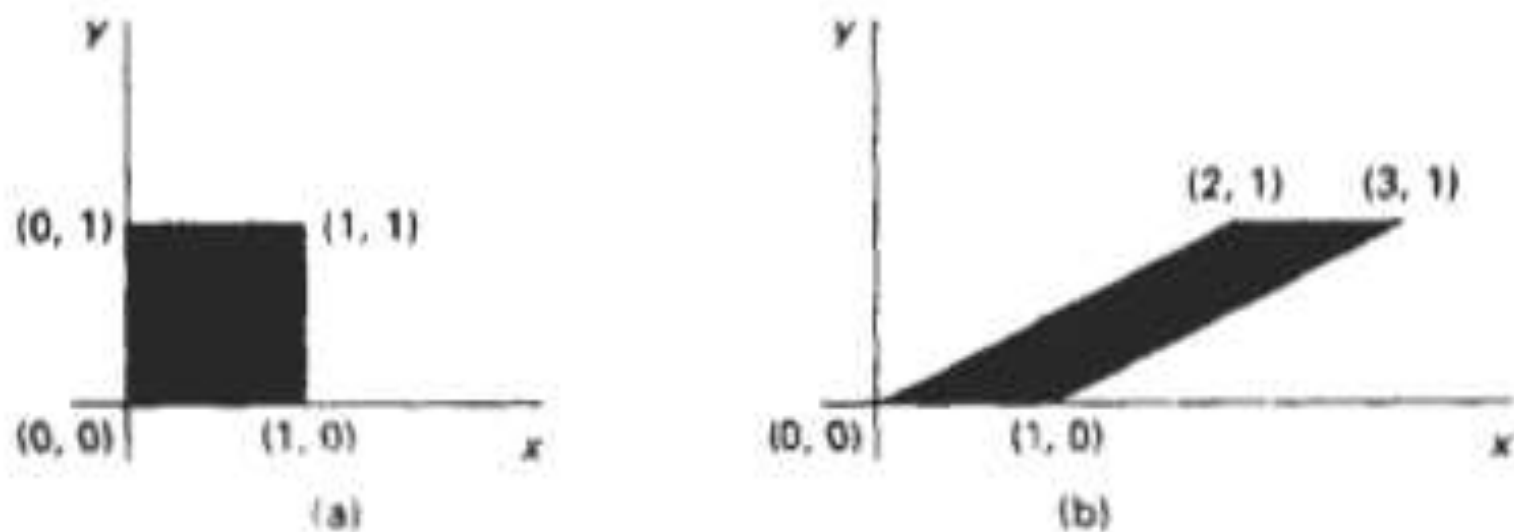


Figure 5-23

A unit square (a) is converted to a parallelogram (b) using the x-direction shear matrix 5-53 with $sh_x = 2$.

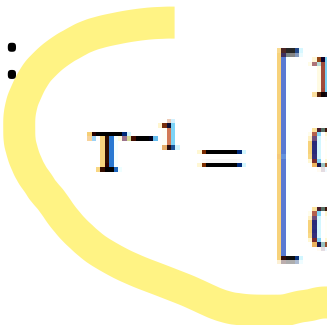
Problems: 2D Geometric Transformations

1. Translate a polygon with coordinates $A(2,5)$, $B(7,10)$ and $C(10,2)$ by 3 units in x direction and 4 units in y direction.
2. A point $(4,3)$ is rotated counterclockwise by an angle 45 degree. Find the rotation matrix and the resultant point.
3. Scale the polygon with coordinate $A(2,5)$, $B(7,10)$ and $C(10,2)$ by 2 units in x direction and 2 units in y direction.

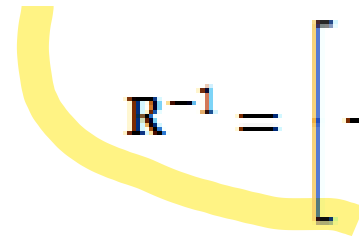
Problems: 2D Homogeneous coordinate

- Give a 3×3 homogeneous coordinate transformation matrix for each of the following translations:
 - Shift the image to the right 3 units
 - Shift the image up 2 units
 - Move the image down 0.5 unit and right 1 unit
 - Move the image down 0.66 unit and left 4 units

Inverse Transformations

- For translation, we obtain the inverse matrix by negating the translation distances.
- Thus, if we have 2D translation distance t_x and t_y , the inverse translation matrix:

$$\mathbf{T}^{-1} = \begin{bmatrix} 1 & 0 & -t_x \\ 0 & 1 & -t_y \\ 0 & 0 & 1 \end{bmatrix}$$
- This produces a translation in the opposite direction, and the product of a translation matrix and its inverse produces the identity matrix.

- An inverse rotation is accomplished by replacing the rotation angle by its negative.


$$\mathbf{R}^{-1} = \begin{bmatrix} \cos \theta & \sin \theta & 0 \\ -\sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Negative values for rotation angles generate rotations in a clockwise direction, so the identity matrix is produced when any rotation matrix is multiplied by its inverse.
- Because only the sine function is affected by the change in sign of the rotation angle, the inverse matrix can also be obtained by interchanging rows and columns.

- Now, we form the inverse matrix for any scaling transformation by replacing the scaling parameters with their reciprocals.
- For 2D scaling with parameters s_x and s_y applied to the coordinate origin, the inverse transformation matrix is:

$$S^{-1} = \begin{bmatrix} \frac{1}{s_x} & 0 & 0 \\ 0 & \frac{1}{s_y} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- The inverse matrix generates an opposite scaling transformation, so the multiplication of any scaling matrix with its inverse produces the identity matrix.


2D Composite Transformations

- With the matrix representations, we can set up a matrix for any sequence of transformations as a composite transformations matrix by calculating the matrix product of the individual transformations.
- Forming products of transformation matrices is often referred to as a concatenation, or composition, of matrices.
- Because a coordinate position is represented with a homogeneous column matrix, we must pre-multiply the column matrix by the matrices representing any transformation sequence.

- Also, because many positions in a scene are typically transformed by the same sequence, it is more efficient to first multiply the transformation matrices to form a single composite matrix.
- Thus, if we want to apply two transformations to point position P , the transformed location would be calculated as:
$$P' = M2 \cdot M1 \cdot P$$
$$= M \cdot P$$

Composite 2D Translations

- If two successive translation vectors $(t1x, t1y)$ and $(t2x, t2y)$ are applied to a two-dimensional coordinate position P , the final transformed location P' is calculated as:


$$\begin{aligned} P' &= T(t2x, t2y) \cdot \{T(t1x, t1y) \cdot P\} \\ &= \{T(t2x, t2y) \cdot T(t1x, t1y)\} \cdot P \end{aligned}$$

- We can verify this result by calculating the matrix product for the two associative groupings.

- Also, the composite transformation matrix for this sequence of translations is:

$$\begin{bmatrix} 1 & 0 & t_{2x} \\ 0 & 1 & t_{2y} \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & t_{1x} \\ 0 & 1 & t_{1y} \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_{1x} + t_{2x} \\ 0 & 1 & t_{1y} + t_{2y} \\ 0 & 0 & 1 \end{bmatrix} \quad \checkmark$$

$$\mathbf{T}(t_{2x}, t_{2y}) \cdot \mathbf{T}(t_{1x}, t_{1y}) = \mathbf{T}(t_{1x} + t_{2x}, t_{1y} + t_{2y})$$

- Which demonstrates that two successive translations are additive.

Composite 2D Rotations

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

$$\begin{aligned} \mathbf{P}' &= \mathbf{R}(\theta_2) \cdot \{\mathbf{R}(\theta_1) \cdot \mathbf{P}\} \\ &= \{\mathbf{R}(\theta_2) \cdot \mathbf{R}(\theta_1)\} \cdot \mathbf{P} \end{aligned}$$

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

$$\mathbf{R}(\theta_2) \cdot \mathbf{R}(\theta_1) = \mathbf{R}(\theta_1 + \theta_2)$$

- so that the final rotated coordinates of a point can be calculated with the composite rotation matrix as:

$$\mathbf{P}' = \mathbf{R}(\theta_1 + \theta_2) \cdot \mathbf{P}$$

Composite 2D Scalings

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

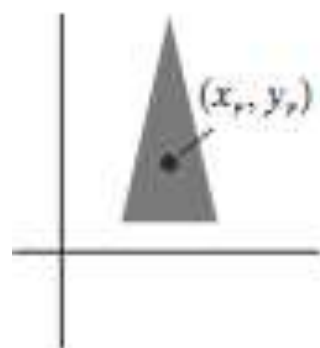
$$\begin{bmatrix} s_{2x} & 0 & 0 \\ 0 & s_{2y} & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} s_{1x} & 0 & 0 \\ 0 & s_{1y} & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} s_{1x} \cdot s_{2x} & 0 & 0 \\ 0 & s_{1y} \cdot s_{2y} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$S(s_{2x}, s_{2y}) \cdot S(s_{1x}, s_{1y}) = S(s_{1x} \cdot s_{2x}, s_{1y} \cdot s_{2y})$$

- The resulting matrix in this case indicates that successive scaling operations are multiplicative.
- That is, if we were to triple the size of an object twice in succession, the final size would be nine times that of the original.

General 2D Pivot-Point Rotation

- When a graphics package provides only a **rotate function with respect to the coordinate origin**, we can generate a two-dimensional rotation about any other **pivot point (xr , yr)** by performing the following sequence of translate-rotate-translate operations:
 1. **Translate** the object so that the **pivot-point position is moved to the coordinate origin**.
 2. **Rotate** the **object** about the coordinate origin.
 3. **Translate** the object so that the **pivot point is returned to its original position**.



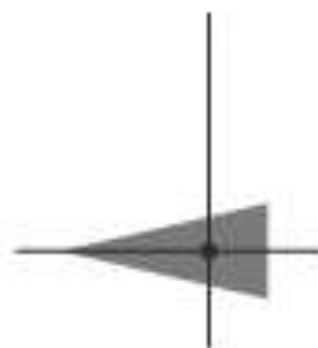
(a)

Original Position
of Object and
Pivot Point



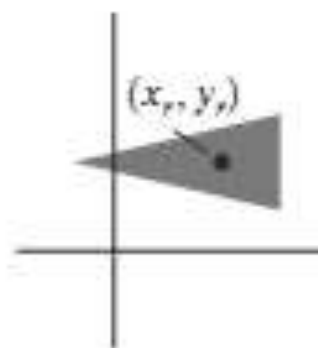
(b)

Translation of
Object so that
Pivot Point
 (x_r, y_r) is at
Origin



(c)

Rotation
about
Origin

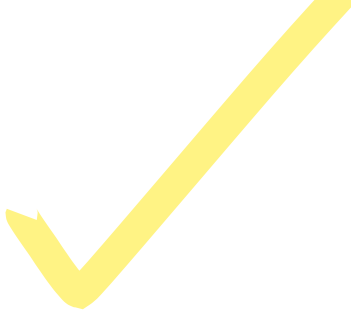


(d)

Translation of
Object so that
the Pivot Point
is Returned
to Position
 (x_r, y_r)

FIGURE 9

A transformation sequence for rotating an object about a specified pivot point using the rotation matrix $\mathbf{R}(\theta)$ of transformation 19.



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

which can be expressed in the form:

$$\mathbf{T}(x_r, y_r) \cdot \mathbf{R}(\theta) \cdot \mathbf{T}(-x_r, -y_r) = \mathbf{R}(x_r, y_r, \theta)$$

General 2D Fixed-Point Scaling

- A transformation sequence to produce a two-dimensional scaling with respect to a selected fixed position (x_f, y_f) , when we have a function that can scale relative to the coordinate origin only. This sequence is:
 1. Translate the object so that the fixed point coincides with the coordinate origin.
 2. Scale the object with respect to the coordinate origin.
 3. Use the inverse of the translation in step (1) to return the object to its original position

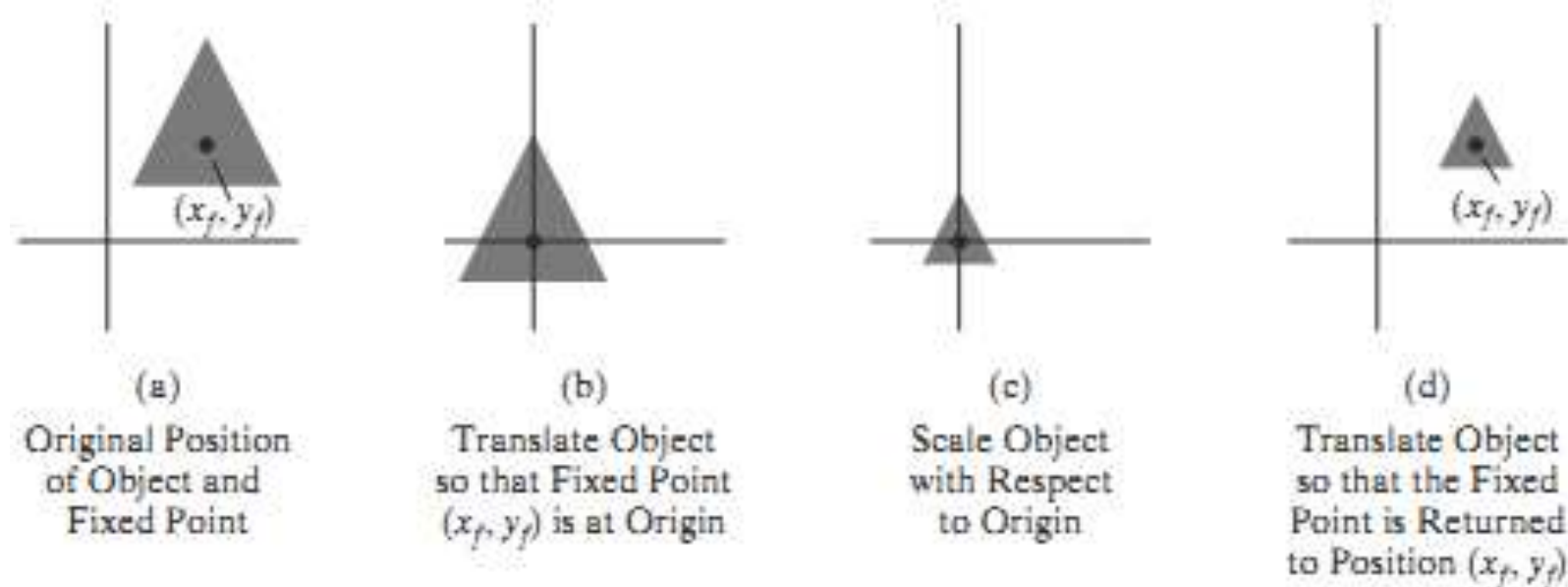


FIGURE 10

A transformation sequence for scaling an object with respect to a specified fixed position using the scaling matrix $\mathbf{S}(s_x, s_y)$ of transformation 21.

- Concatenating the matrices for these three operations produces the required scaling matrix:

$$\begin{bmatrix} 1 & 0 & x_f \\ 0 & 1 & y_f \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & -x_f \\ 0 & 1 & -y_f \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & x_f(1-s_x) \\ 0 & s_y & y_f(1-s_y) \\ 0 & 0 & 1 \end{bmatrix}$$

OR

$$\mathbf{T}(x_f, y_f) \cdot \mathbf{S}(s_x, s_y) \cdot \mathbf{T}(-x_f, -y_f) = \mathbf{S}(x_f, y_f, s_x, s_y)$$

- This transformation is generated automatically in systems that provide a scale function that accepts coordinates for the fixed point.

General 2D Scaling Directions

- Parameters s_x and s_y scale objects along the x and y directions.
- We can **scale an object in other directions** by rotating the object to align the desired scaling directions with the coordinate axes before applying the scaling transformation.
- Suppose we want to apply scaling factors with values specified by parameters s_1 and s_2 in the directions.

- To accomplish the scaling without changing the orientation of the object, we first perform a rotation so that the directions for s_1 and s_2 coincide with the x and y axes, respectively.
- Then the scaling transformation $S(s_1, s_2)$ is applied, followed by an opposite rotation to return points to their original orientations.

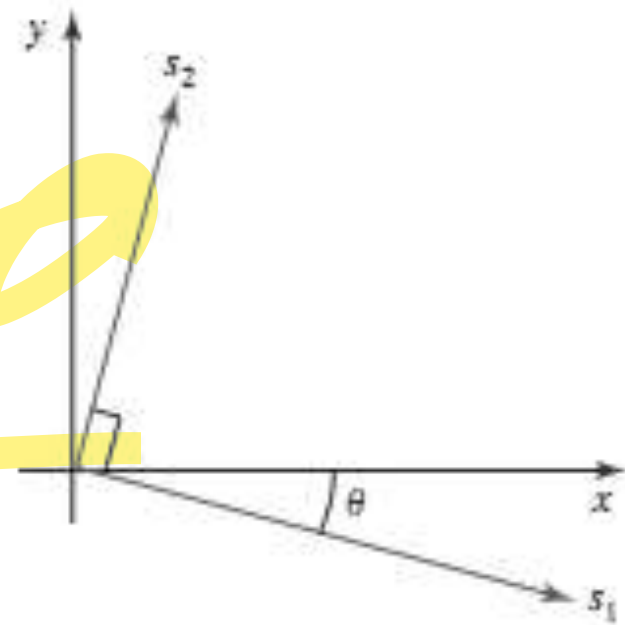


FIGURE 11

Scaling parameters s_1 and s_2 along orthogonal directions defined by the angular displacement θ .

- The composite matrix resulting from the product of these three transformations is

$$\mathbf{R}^{-1}(\theta) \cdot \mathbf{S}(s_1, s_2) \cdot \mathbf{R}(\theta) = \begin{bmatrix} s_1 \cos^2 \theta + s_2 \sin^2 \theta & (s_2 - s_1) \cos \theta \sin \theta & 0 \\ (s_2 - s_1) \cos \theta \sin \theta & s_1 \sin^2 \theta + s_2 \cos^2 \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

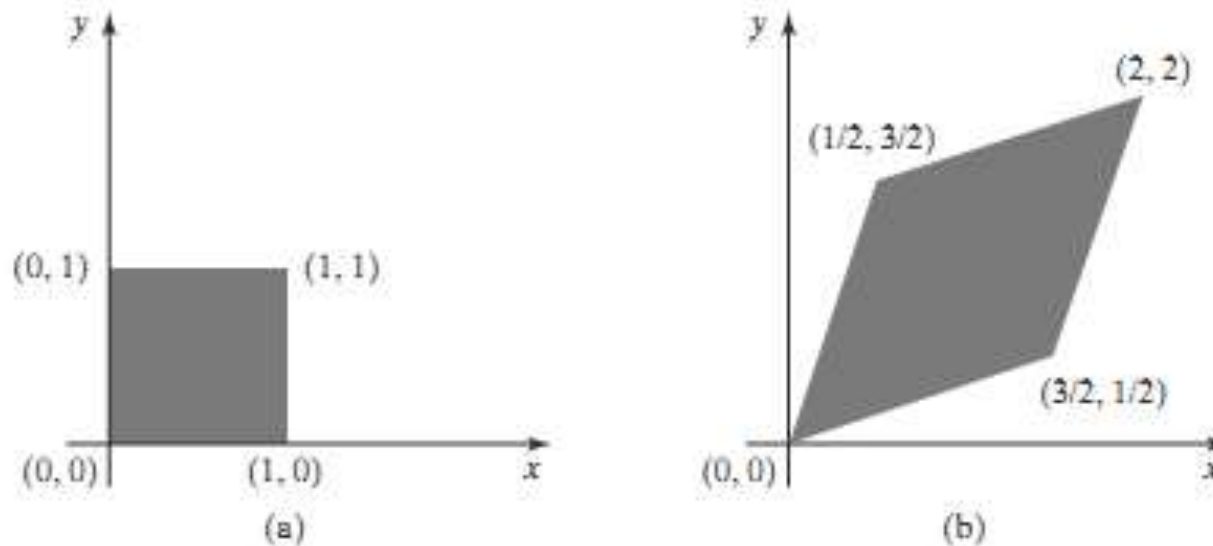


FIGURE 12

A square (a) is converted to a parallelogram (b) using the composite transformation matrix 39, with

$s_1 = 1$, $s_2 = 2$, and $\theta = 45^\circ$.

Problems

- Show how shear transformation may be expressed in terms of rotation and scaling.
- Apply the shearing transformation to square with $A(0,0)$, $B(1,0)$, $C(1,1)$ and $D(0,1)$ as given below:
 - Shear parameter value of 0.5 relative to the line $y_{ref} = -1$
 - Shear parameter value of 0.5 relative to the line $x_{ref} = -1$

2-Dimensional Viewing and Clipping

- A graphics package allows a user to specify which part of a defined picture is to be displayed and where that part is to be placed on the display device.
- Any convenient Cartesian coordinate system, referred to as the world-coordinate reference frame, can be used to define the picture.
- For a 2D picture, a view is selected by specifying a region of the xy plane that contains the total picture or any part of it.
- The picture parts within the selected areas are then mapped onto specified areas of the device coordinates.
- When multiple view areas are selected, these areas can be placed in separate display locations, or some areas could be inserted into other, larger display areas.

1. The 2D Viewing Pipeline

- A section of a two-dimensional scene that is selected for display is called a **clipping window** because all parts of the scene outside the selected section are “clipped” off.
- The only part of the scene that shows up on the screen is what is inside the clipping window.
- Sometimes the **clipping window** is alluded to as the **world window** or the **viewing window**.

- Graphics packages allow us also to control the placement within the display window using another “window” called the viewport.
- Objects inside the clipping window are mapped to the viewport, and it is the viewport that is then positioned within the display window.
- The clipping window selects what we want to see; the viewport indicates where it is to be viewed on the output device.
- By changing the position of a viewport, we can view objects at different positions on the display area of an output device.
- Multiple viewports can be used to display different sections of a scene at different screen positions.
- Usually, clipping windows and viewports are rectangles in standard position, with the rectangle edges parallel to the coordinate axes.

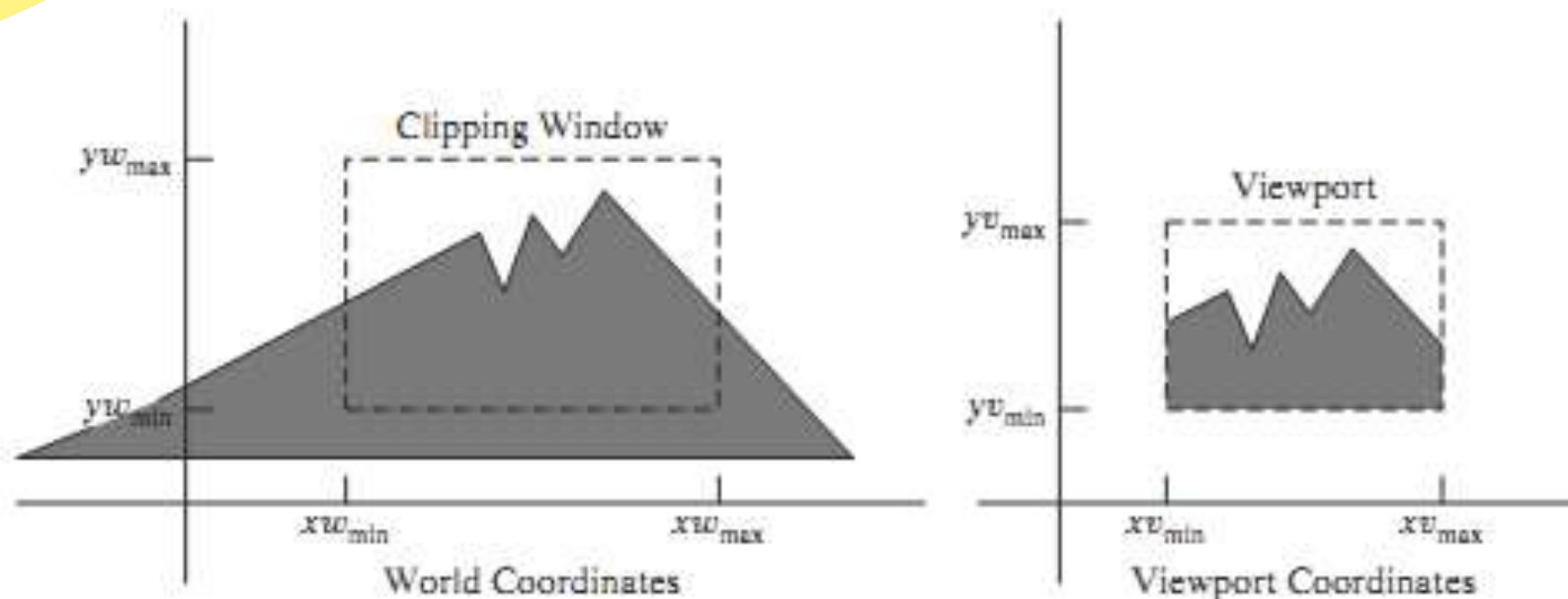


FIGURE 1

A clipping window and associated viewport, specified as rectangles aligned with the coordinate axes.

- The mapping of a two-dimensional, world-coordinate scene description to device coordinates is called a two-dimensional viewing transformation. Sometimes this transformation is simply referred to as the window-to-viewport transformation or the windowing transformation.

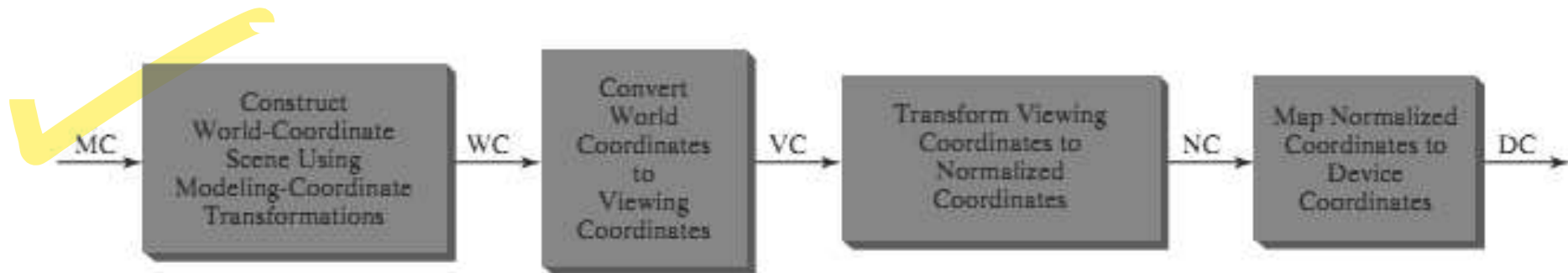


FIGURE 2
Two-dimensional viewing-transformation pipeline.

- Once a world-coordinate scene has been constructed, we could set up a separate two-dimensional, viewing-coordinate reference frame for specifying the clipping window.
- But the clipping window is often just defined in world coordinates, so viewing coordinates for two-dimensional applications are the same as world coordinates.
- Clipping is usually performed in normalized coordinates.
- This allows us to reduce computations by first concatenating the various transformation matrices.
- Clipping procedures are of fundamental importance in computer graphics.

2. The Clipping Window

- To achieve a particular viewing effect in an application program, we could design our own clipping window with any shape, size, and orientation we choose.
- The simplest window edges to clip against are straight lines that are parallel to the coordinate axes.
- Therefore, graphics packages commonly allow only rectangular clipping windows aligned with the x and y axes.
- Rectangular clipping windows in standard position are easily defined by giving the coordinates of two opposite corners of each rectangle.

Viewing-Coordinate Clipping Window

- A general approach to the two-dimensional viewing transformation is to set up a viewing-coordinate system within the world-coordinate frame.
- This viewing frame provides a reference for specifying a rectangular clipping window with any selected orientation and position.
- To obtain a view of the world-coordinate scene as determined by the clipping window of, we just need to transfer the scene description to viewing coordinates.

- We choose an origin for a two-dimensional viewing-coordinate frame at some world position $P_0 = (x_0, y_0)$, and we can establish the orientation using a world vector V that defines the y_{view} direction. Vector V is called the two-dimensional view up vector.

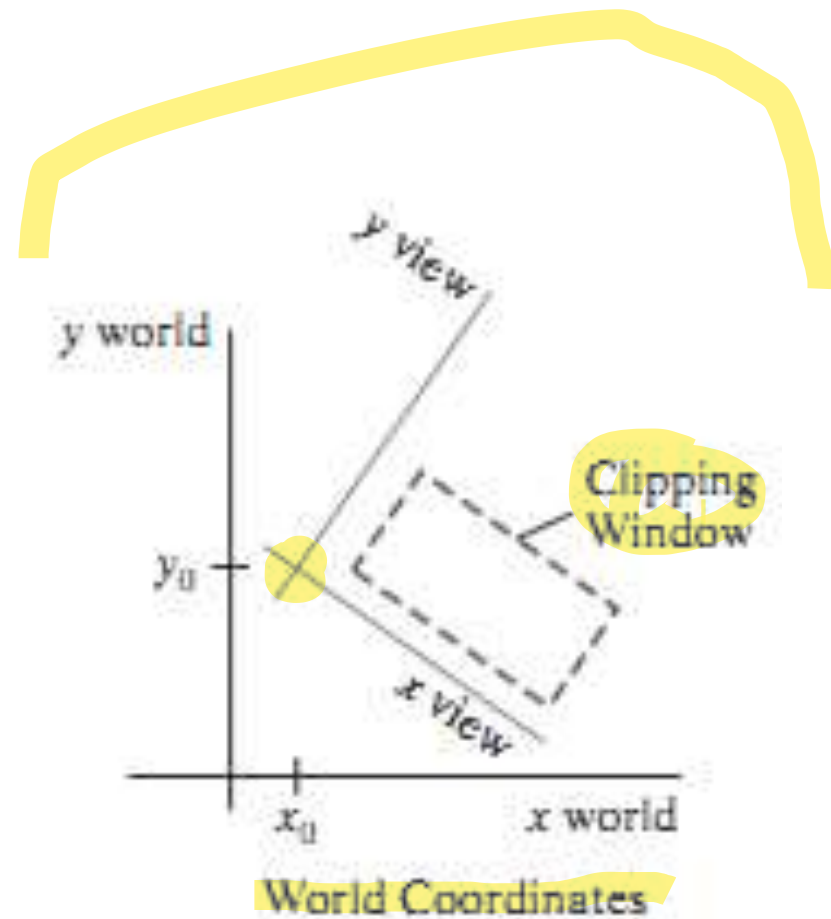


FIGURE 3

A rotated clipping window defined in viewing coordinates.

By changing the position of the viewport

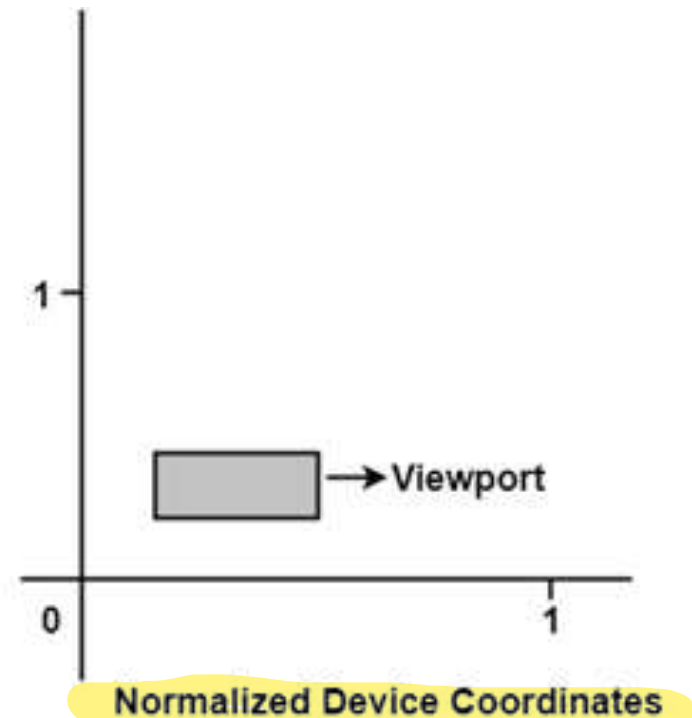
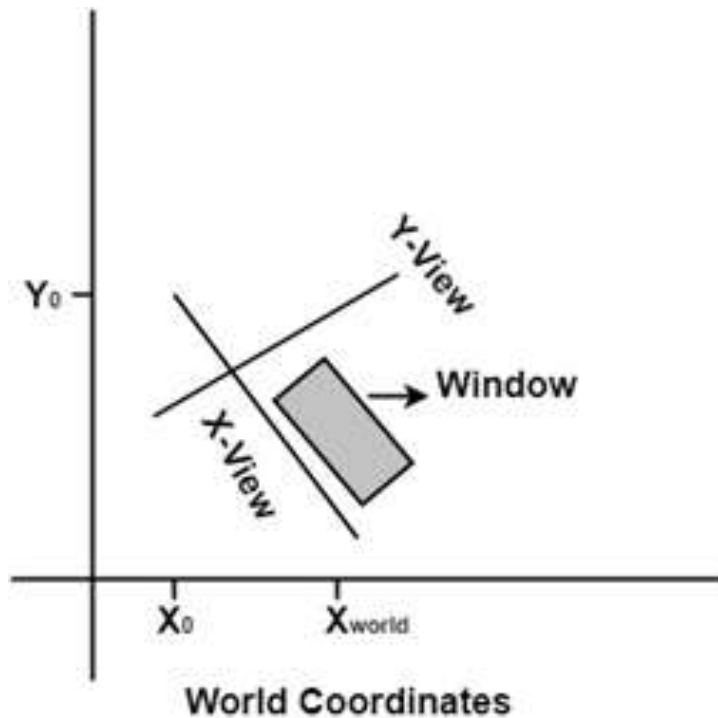


Fig: Setting up a rotated world window and corresponding normalized coordinate viewport.

- Once we have established the parameters that defines the viewing-coordinate frame, we transform the scene description to the viewing system.
- The first step in the transformation sequence is to translate the viewing origin to the world origin.
- Next, we rotate the viewing system to align it with the world frame.
- Object positions in world coordinates are then converted to viewing coordinates with the composite two-dimensional transformation matrix:

$$M_{wc,vc} = R \cdot T$$

World-Coordinate Clipping Window

- A routine for defining a standard, rectangular clipping window in world coordinates is typically provided in a graphics-programming library.
- We simply specify two world-coordinate positions, which are then assigned to the two opposite
- Once the clipping window has been established, the scene description is processed through the viewing routines to the output device.

Imp

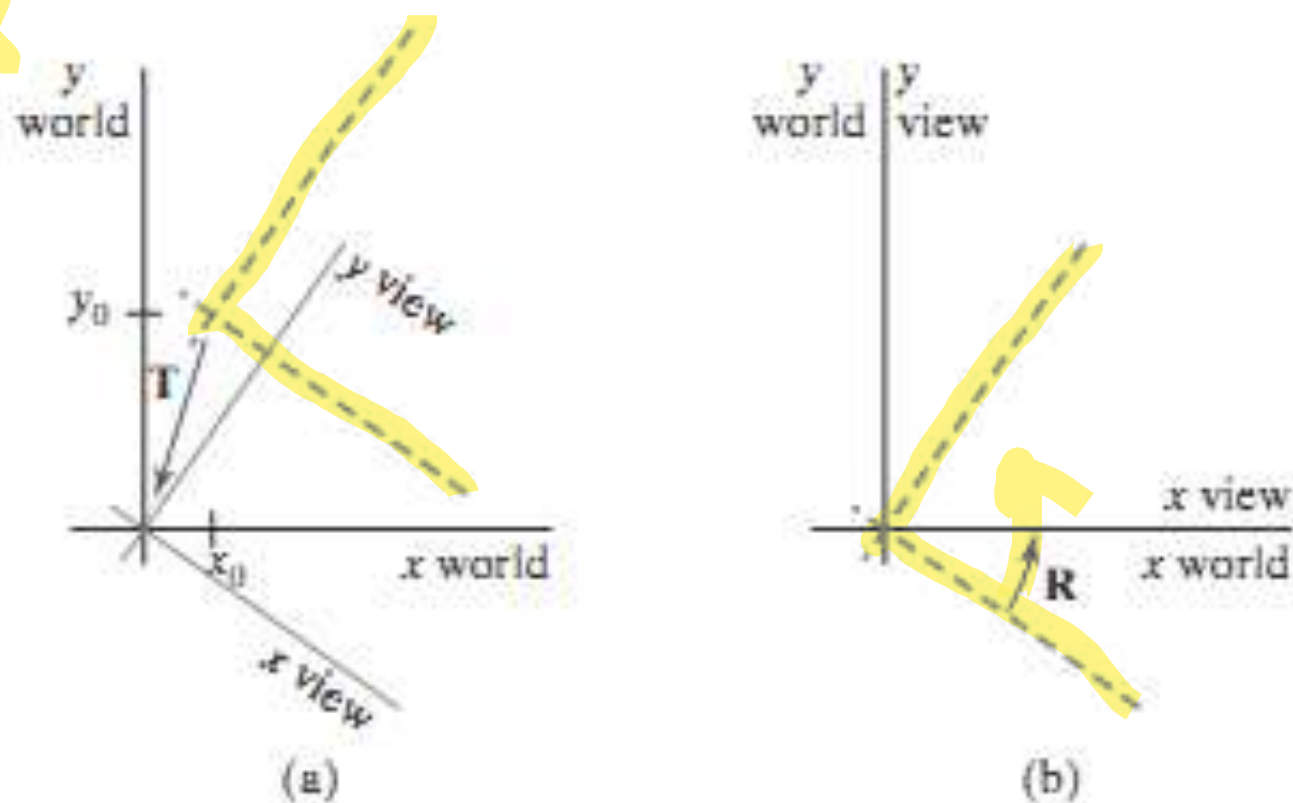


FIGURE 4

A viewing-coordinate frame is moved into coincidence with the world frame by (a) applying a translation matrix T to move the viewing origin to the world origin, then (b) applying a rotation matrix R to align the axes of the two systems.

- If we want to obtain a rotated view of a two-dimensional scene, as discussed in the previous section, we perform exactly the same steps as described there, but without considering a viewing frame of reference.
- Thus, we simply rotate (and possibly translate) objects to a desired position and set up the clipping window—all in world coordinates.

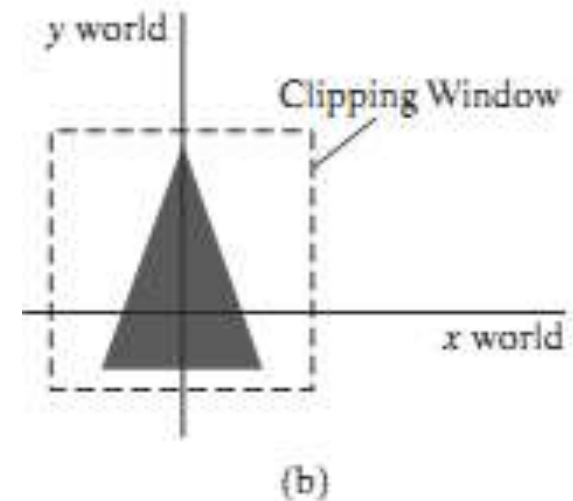
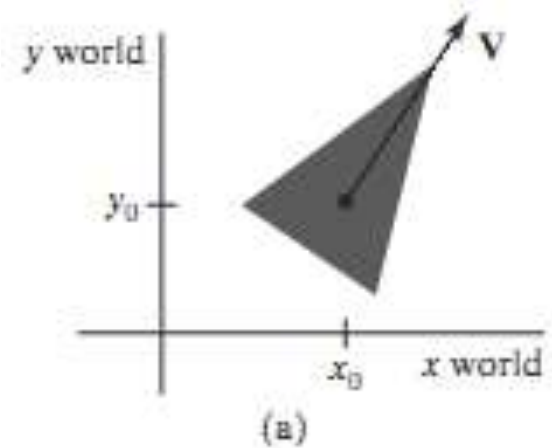


FIGURE 5

A triangle (a), with a selected reference point and orientation vector, is translated and rotated to position (b) within a clipping window.

By varying the size of the viewports

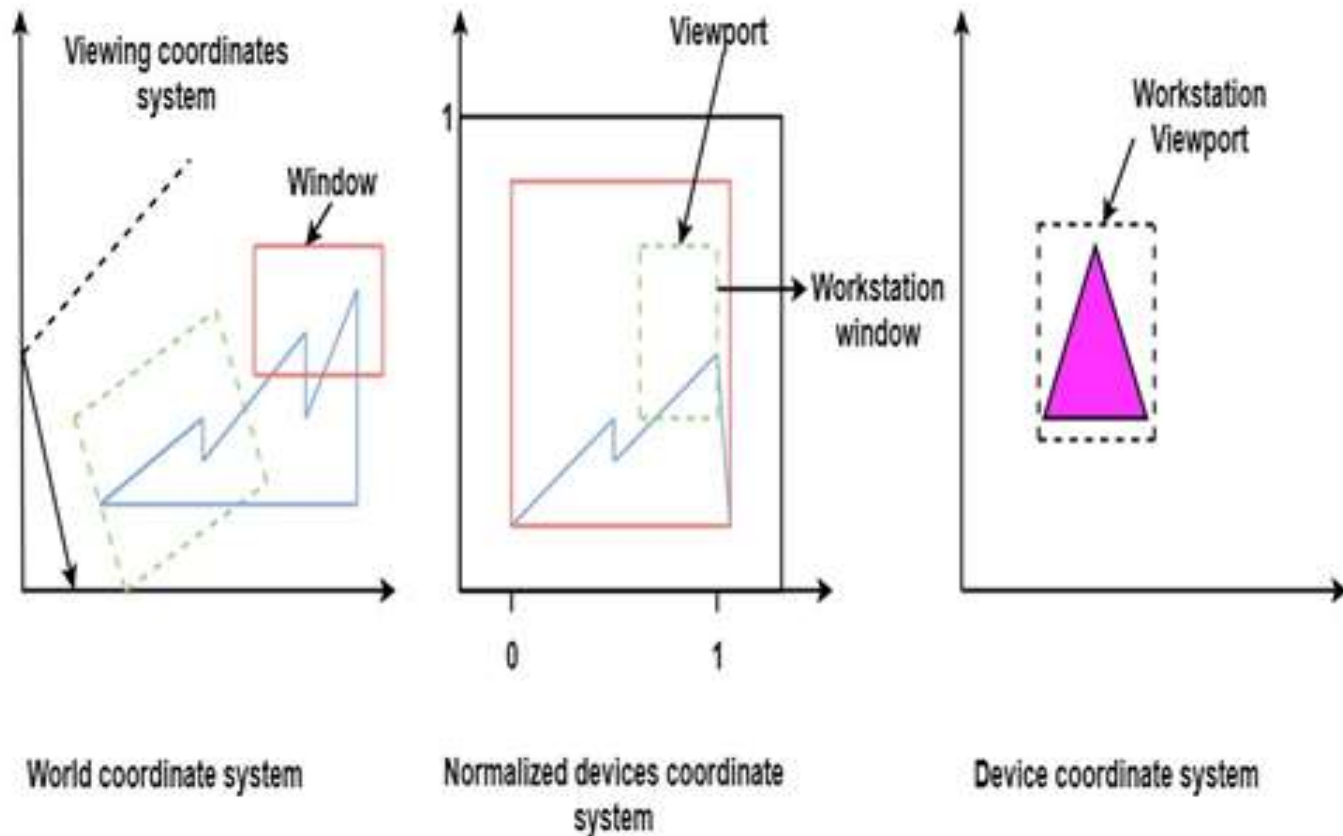


Fig: Zooming effects by mapping different-sized windows on a fixed-size viewport.

3. Normalization and Viewport Transformations

- With some graphics packages, the normalization and window-to-viewport transformations are combined into one operation.
- In this case, the viewport coordinates are often given in the range from 0 to 1 so that the viewport is positioned within a unit square.
- After clipping, the unit square containing the viewport is mapped to the output display device.

Mapping the Clipping Window into a Normalized Viewport

- To illustrate the general procedures for the normalization and viewport transformations, we first consider a viewport defined with normalized coordinate values between 0 and 1.
- Object descriptions are transferred to this normalized space using a transformation that maintains the same relative placement of a point in the viewport as it had in the clipping window.
- If a coordinate position is at the center of the clipping window, for instance, it would be mapped to the center of the viewport.

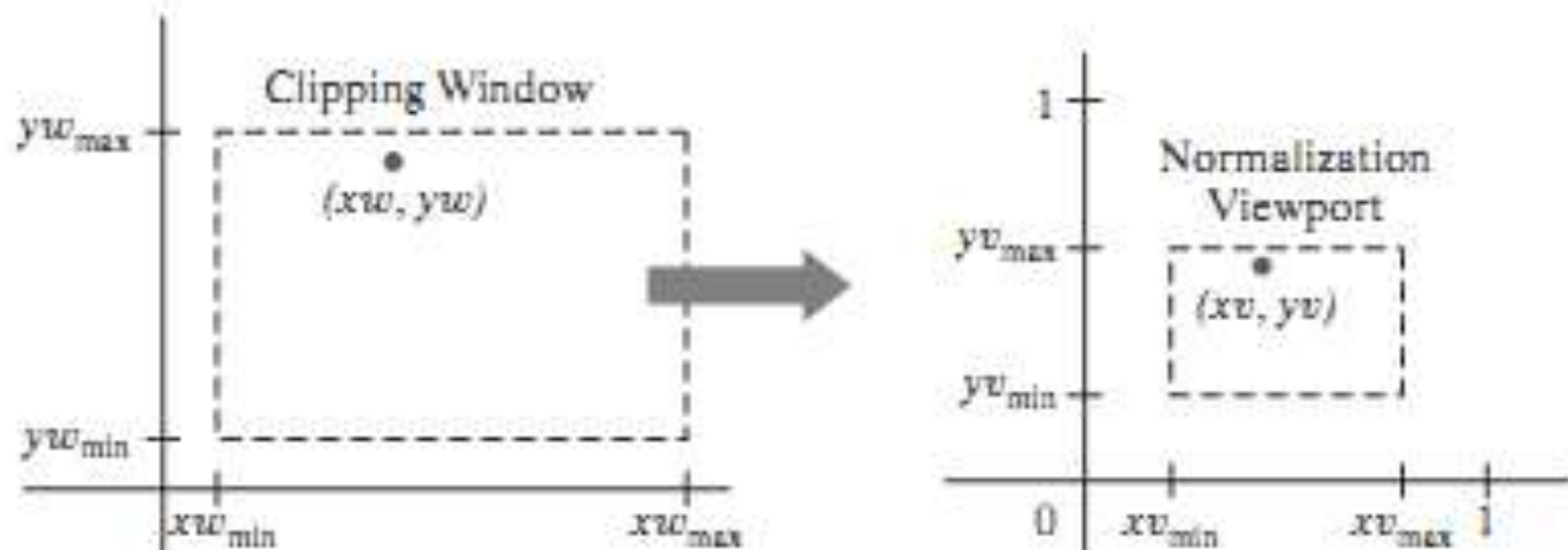


FIGURE 6

A point (xw, yw) in a world-coordinate clipping window is mapped to viewport coordinates (xv, yv) , within a unit square, so that the relative positions of the two points in their respective rectangles are the same.

- Position (x_w, y_w) in the clipping window is mapped to position (x_v, y_v) in the associated viewport.
- To transform the world-coordinate point into the same relative position within the viewport, we require that:

$$\frac{x_v - x_{v_{\min}}}{x_{v_{\max}} - x_{v_{\min}}} = \frac{x_w - x_{w_{\min}}}{x_{w_{\max}} - x_{w_{\min}}}$$

$$\frac{y_v - y_{v_{\min}}}{y_{v_{\max}} - y_{v_{\min}}} = \frac{y_w - y_{w_{\min}}}{y_{w_{\max}} - y_{w_{\min}}}$$

$$s = \frac{V}{W}$$

$$x_v = s_x x_w + t_x$$

$$y_v = s_y y_w + t_y$$

- and the translation factors are:

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

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

- Because we are simply mapping world-coordinate positions into a viewport that is positioned near the world origin, we can also derive using any transformation sequence that converts the rectangle for the clipping window into the viewport rectangle.

Mapping the Clipping Window into a Normalized Square

- Another approach to two-dimensional viewing is to transform the clipping window into a normalized square, clip in normalized coordinates, and then transfer the scene description to a viewport specified in screen coordinates.

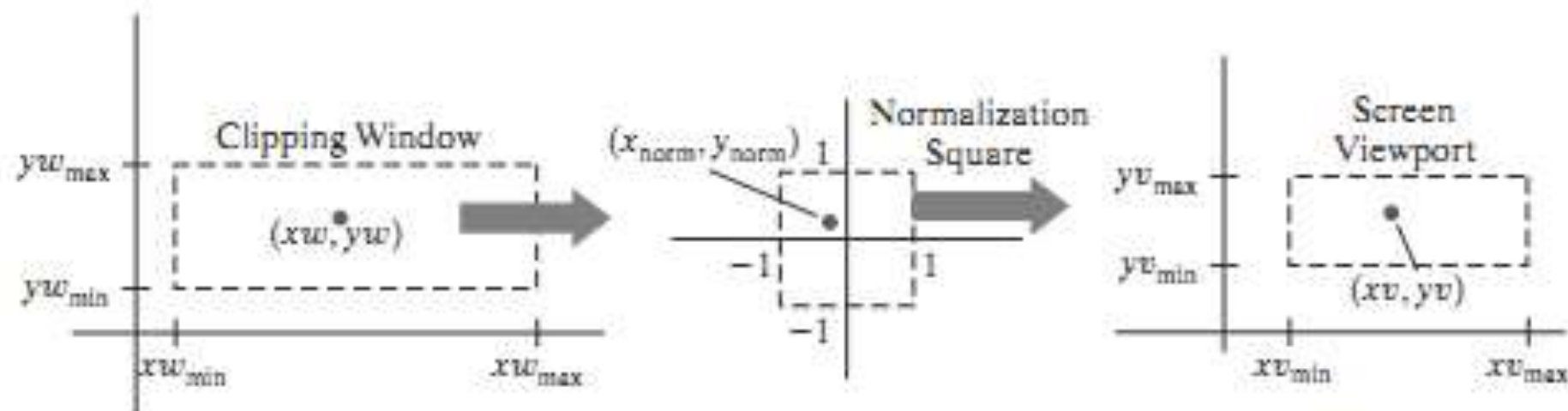


FIGURE 7

A point (x_w, y_w) in a clipping window is mapped to a normalized coordinate position (x_{norm}, y_{norm}) , then to a screen-coordinate position (x_v, y_v) in a viewport. Objects are clipped against the normalization square before the transformation to viewport coordinates occurs.

- Making these substitutions in the expressions for tx , ty , sx , and sy , we have:

$$M_{\text{window, normsquare}} = \begin{bmatrix} \frac{2}{xw_{\max} - xw_{\min}} & 0 & -\frac{xw_{\max} + xw_{\min}}{xw_{\max} - xw_{\min}} \\ 0 & \frac{2}{yw_{\max} - yw_{\min}} & -\frac{yw_{\max} + yw_{\min}}{yw_{\max} - yw_{\min}} \\ 0 & 0 & 1 \end{bmatrix}$$

- Similarly, after the clipping algorithms have been applied, the normalized square with edge length equal to 2 is transformed into a specified viewport.

$$\mathbf{M}_{\text{normsquare, viewport}} = \begin{bmatrix} \frac{xv_{\max} - xv_{\min}}{2} & 0 & \frac{xv_{\max} + xv_{\min}}{2} \\ 0 & \frac{yv_{\max} - yv_{\min}}{2} & \frac{yv_{\max} + yv_{\min}}{2} \\ 0 & 0 & 1 \end{bmatrix}$$

- The last step in the viewing process is to position the viewport area in the display window.
- Typically, the lower-left corner of the viewport is placed at a coordinate position specified relative to the lower-left corner of the display window.

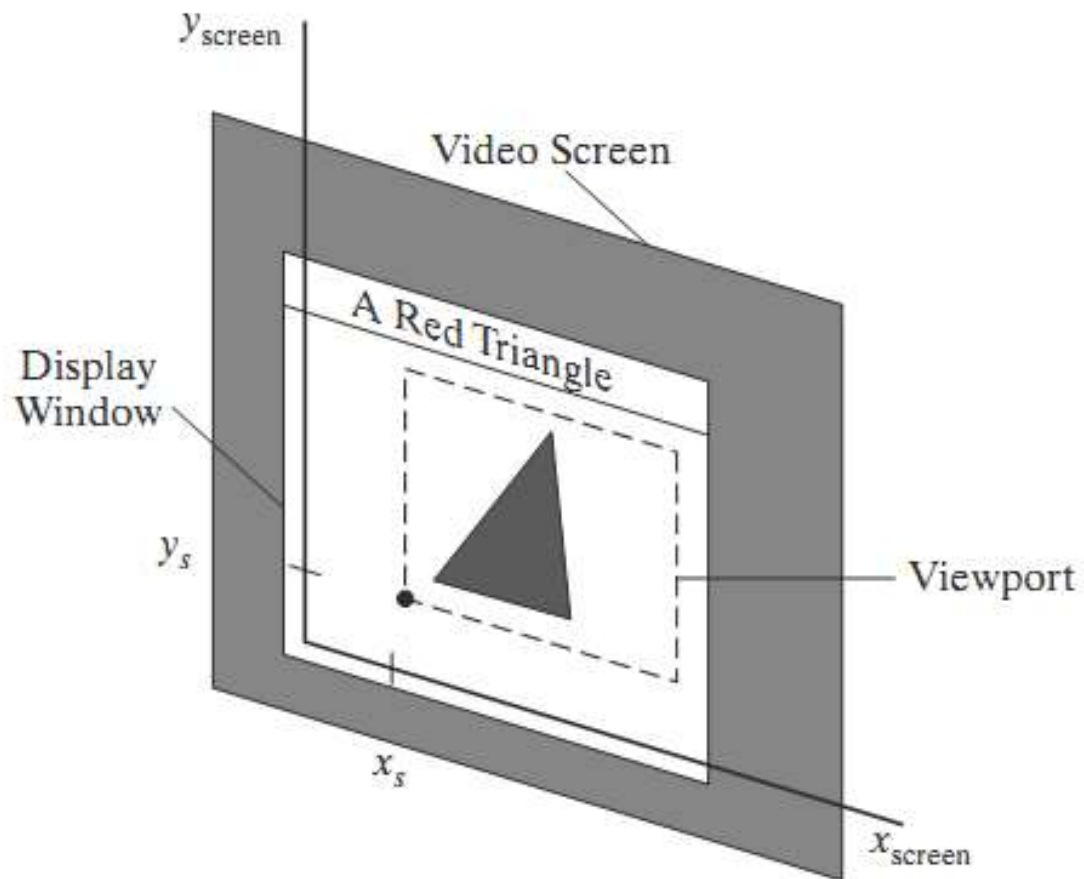


FIGURE 8

A viewport at coordinate position (x_s, y_s) within a display window.

Problems

1. Show that 2D reflection through X axis followed by 2D reflection through the line $Y = -X$ is equivalent to a pure rotation about the origin.
2. Prove that successive 2D rotations are additive, i.e. $R(\theta_1) \cdot R(\theta_2) = R(\theta_1 + \theta_2)$
3. Prove that 2D rotation and scaling commute if $S_x = S_y$ or $\theta = n\pi$ for integral n and that otherwise they do not.