Computer Graphics

2D Transformations

3D Transformations

By

Kareem Ahmed

2D-Translation Matrix

- Using Homogeneous-coordinate approach, we can represent the equations for a 2D translation of a coordinate position using the following matrix multiplication.

$$\begin{bmatrix} x` \\ y` \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- This translation operation can be written in the abbreviated form

$$P` = T(t_x , t_y) . P$$

2D-Rotation Matrix

- Using Homogeneous-coordinate approach, we can represent the equations for a 2D rotation about the coordinate origin in the matrix form:

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

- This rotation transformation operation can be written in the abbreviated form

$$P` = R(\theta).P$$

2D-Rotation Matrix

- In some graphics libraries, a 2D rotation function generates only rotation about the coordinate origin. A rotation about any other pivot point must then be performed as a sequence of transformation operations.
- An alternative approach in a graphics package is to provide additional parameters in the rotation routine for the pivot-point coordinates. A rotation routine that includes a pivot-point parameter then sets up a general rotation matrix without the need to invoke a succession of transformation functions.

2D-Scaling Matrix

- Using Homogeneous-coordinate approach, we can represent the equations for a 2D scaling relative to the coordinate origin in the matrix form:

$$\begin{bmatrix} x` \\ y` \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- This Scaling transformation operation can be written in the abbreviated form

$$P` = S(S_x, S_y) . P$$

2D-Scaling Matrix

- Some libraries provide a scaling function that can generate only scaling with respect to the coordinate origin. In this case, a scaling transformation relative to another reference position is handled as a succession of transformation operations.
- However, other systems do include a general scaling routine that can construct the homogeneous matrix for scaling with respect to a designated fixed point.

Inverse Transformations

Inverse 2D-translation Matrix

- If we have 2D translation distances t_x and t_y, the inverse 2D translation homogeneous matrix is:

$$T^{-1} = \begin{bmatrix} 1 & 0 & -t_x \\ 0 & 1 & -t_y \\ 0 & 0 & 1 \end{bmatrix}$$

- This produces a translation in the opposite direction.
- The product of a translation matrix and its inverse produces the identity matrix.

Inverse 2D-Rotation Matrix

- An inverse rotation is accomplished by replacing the rotation angle by its negative.
- For example, a 2D rotation through an angle θ about the coordinate origin has the inverse transformation matrix.

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

Inverse 2D-Rotation Matrix

- Negative values for rotation angles generate rotations in a clockwise direction.
- The identity matrix is produced when any rotation matrix is multiplied by its inverse.
- $R^{-1} = R^{T}$

Inverse 2D-Scaling Matrix

- The inverse matrix for any scaling transformation is obtained by replacing the scaling parameters with their reciprocals.
- For 2D scaling with parameters S_x and S_y applied relative 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.

Two-Dimensional Composite Transformations

Two-Dimensional Composite Transformations

- Using matrix representations, we can set up a sequence of transformations as a **composite transformation matrix** by calculating the product of the individual transformations.
- 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

$$P` = M_2 \cdot M_1 \cdot P$$
$$= M \cdot P$$

The coordinate position is transformed using the composite matrix M, rather than applying the individual transformations M_1 and then M_2.

Composite 2D Translation

- If two successive translation vectors (t_{1x}, t_{1y}) and (t_{2x}, t_{2y}) are applied to a 2D coordinate position P, the final transformed location $P`$ is calculated as:

$$P` = T(t_{2x}, t_{2y}) \cdot [T(t_{1x}, t_{1y}) \cdot P]$$
$$P` = [T(t_{2x}, t_{2y}) \cdot T(t_{1x}, t_{1y})] \cdot P$$

- The composite transformation matrix for this sequence of translations is

$$\begin{bmatrix} 1 & 0 & t_{2x} \\ 0 & 1 & t_{2y} \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & t_{1x} \\ 0 & 1 & t_{1y} \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_{1x} + t_{2x} \\ 0 & 1 & t_{1y} + t_{2y} \\ 0 & 0 & 1 \end{bmatrix}$$
$$T(t_{2x}, t_{2y}) \cdot T(t_{1x}, t_{1y}) = T(t_{1x} + t_{2x}, t_{1y} + t_{2y})$$

- Which demonstrates that two successive translations are additive

Composite 2D Rotation

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

$$P` = R(\theta_2) \cdot [R(\theta_1) \cdot P]$$
$$P` = [R(\theta_2) \cdot R(\theta_1)] \cdot P$$

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

$$R(\theta_2) \cdot R(\theta_1) = R(\theta_1 + \theta_2)$$

Composite 2D Rotation

$$\begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos\theta_2 & -\sin\theta_2 & 0 \\ \sin\theta_2 & \cos\theta_2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \cos\theta_1\cos\theta_2 - \sin\theta_1\sin\theta_2 & -\cos\theta_1\sin\theta_2 - \sin\theta_1\cos\theta_2 & 0 \\ \sin\theta_1\cos\theta_2 + \cos\theta_1\sin\theta_2 & -\sin\theta_1\sin\theta_2 + \cos\theta_1\cos\theta_2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \cos(\theta_1+\theta_2) & -\sin(\theta_1+\theta_2) & 0 \\ \sin(\theta_1+\theta_2) & \cos(\theta_1+\theta_2) & 0 \\ 0 & 0 & 1 \end{bmatrix} = R(\theta_1+\theta_2)$$

Composite 2D Scaling

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

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

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

- 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 successive, the final size would be nine times that of the original.

2D Pivot-Point Rotation

- The 2D rotation about any pivot point (x_r, y_r) can be performed by 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.

2D Pivot-Point Rotation

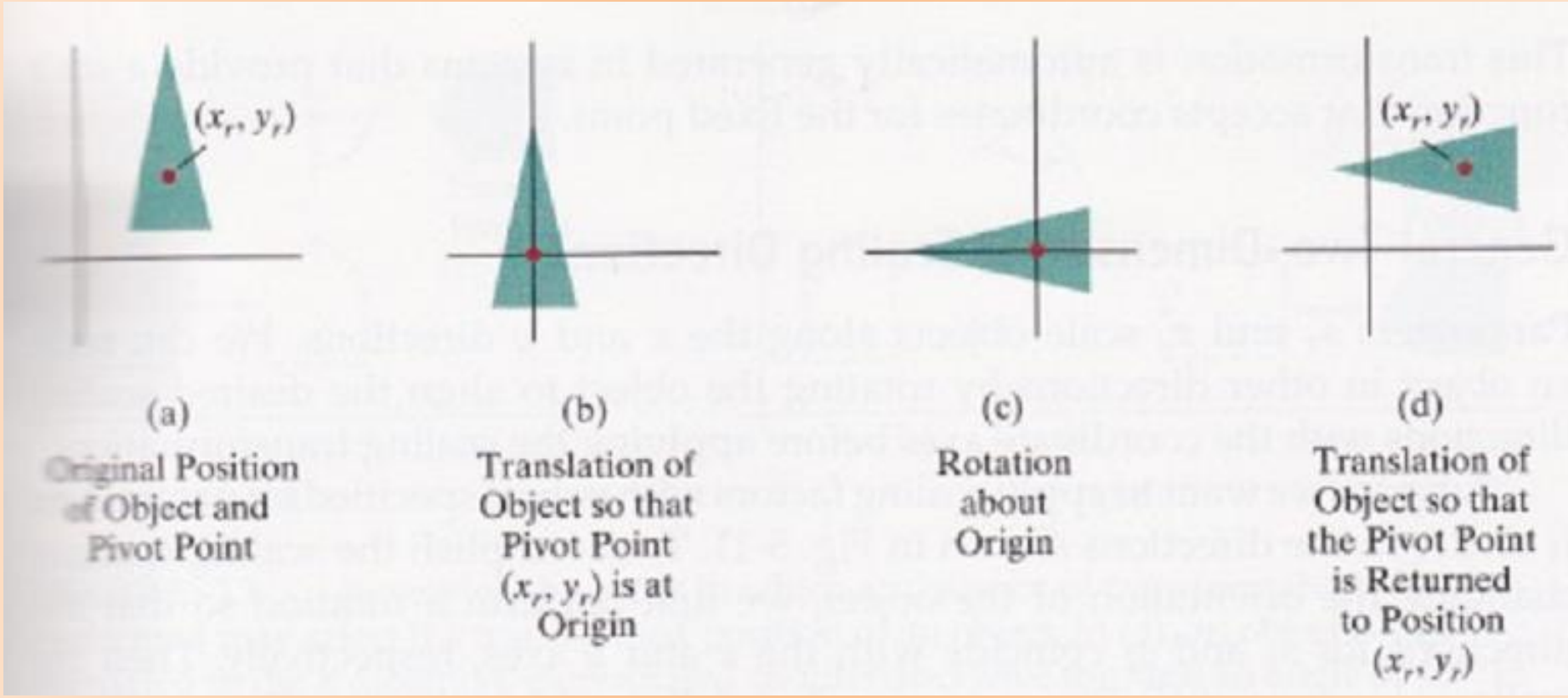
$$\begin{bmatrix} 1 & 0 & x_r \\ 0 & 1 & y_r \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & -x_r \\ 0 & 1 & -y_r \\ 0 & 0 & 1 \end{bmatrix}$$

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

- Which can be expressed in the form

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

2D Pivot-Point Rotation



2D Fixed-Point Scaling

- To produce a 2D scaling with respect to a selected fixed position (x_f, y_f), the ***translate-scale-translate*** sequence is used:
  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.

2D Fixed-Point Scaling

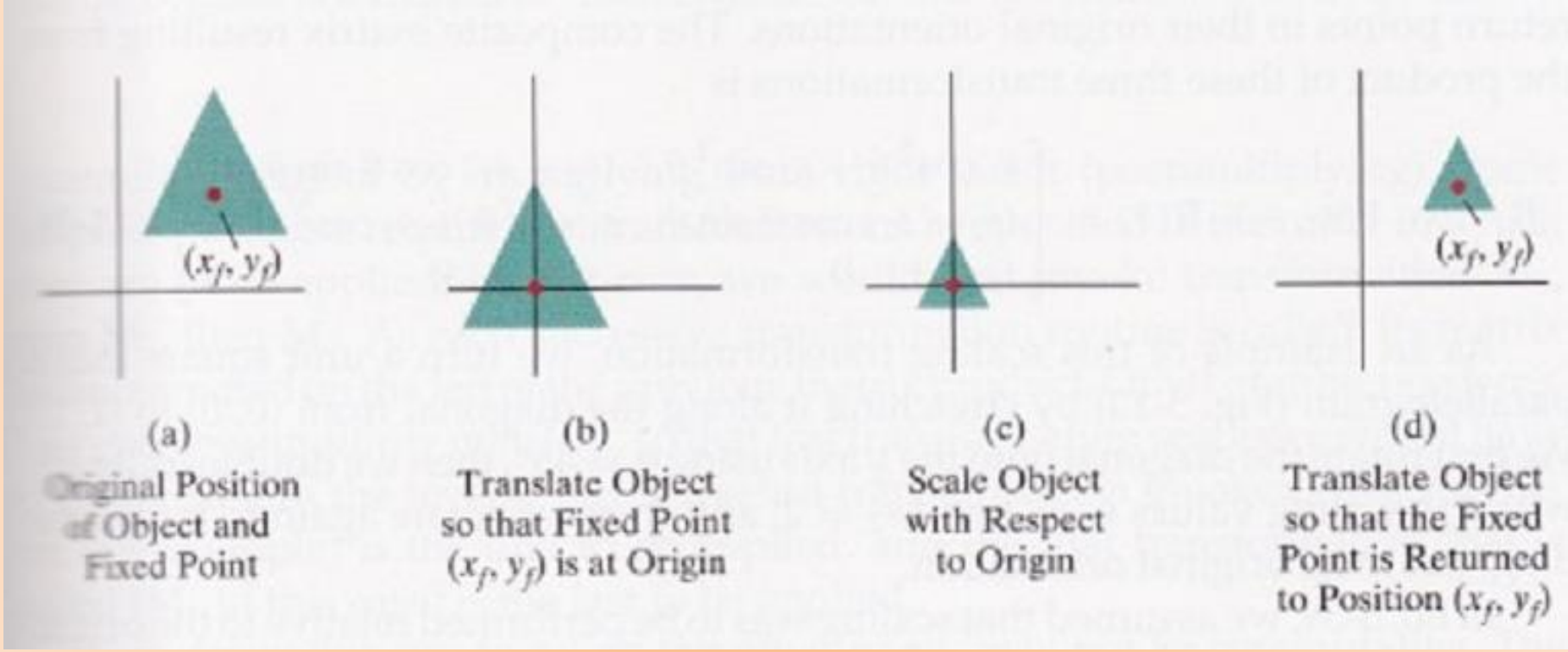
$$\begin{bmatrix} 1 & 0 & x_f \\ 0 & 1 & y_f \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & -x_f \\ 0 & 1 & -y_f \\ 0 & 0 & 1 \end{bmatrix}$$

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

- Which can be expressed in the form

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

2D Fixed-Point Scaling



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.

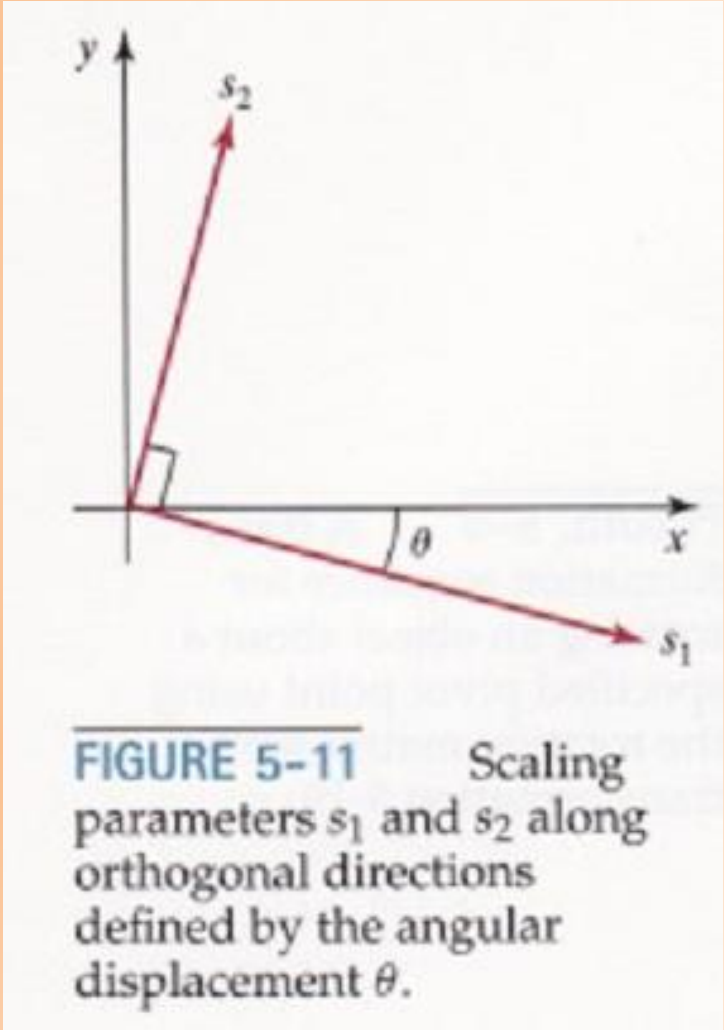
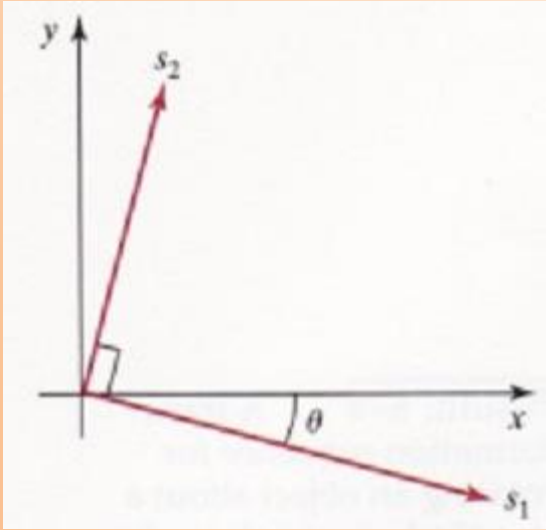


FIGURE 5-11 Scaling parameters s_1 and s_2 along orthogonal directions defined by the angular displacement θ.

General 2D Scaling Directions

- Suppose we want to apply scaling factors with values specified by parameters s_1 and s_2 in the directions shown in this figure.



- To accomplish this process:
  - 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.

General 2D Scaling Directions

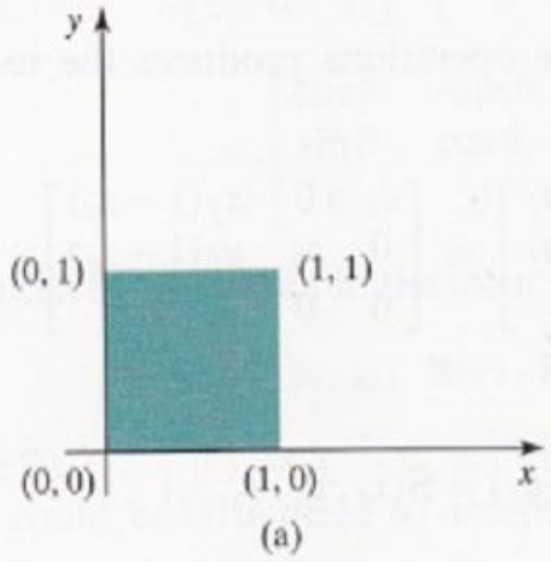
- The composite matrix resulting from the product of these transformations is

$$R^{-1}(\theta) \cdot \mathrm{S}(s_1, s_2) \cdot R(\theta) =$$

$$\begin{bmatrix} s_1 cos^2\theta + s_2 sin^2\theta & (s_2 - s_1) cos\theta sin\theta & 0 \\ (s_2 - s_1) cos\theta sin\theta & s_1 sin^2\theta + s_2 cos^2\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

General 2D Scaling Directions

- An example is to turn a unit square into a parallelogram by stretching it along the diagonal form (0,0) to (1,1).
- We first rotate the diagonal onto the y axis using $\theta = 45^{\circ}$, then we double its length with the scaling factors $s_1 = 1$ and $s_2 = 2$, and then we rotate again to return the diagonal to its original orientation.



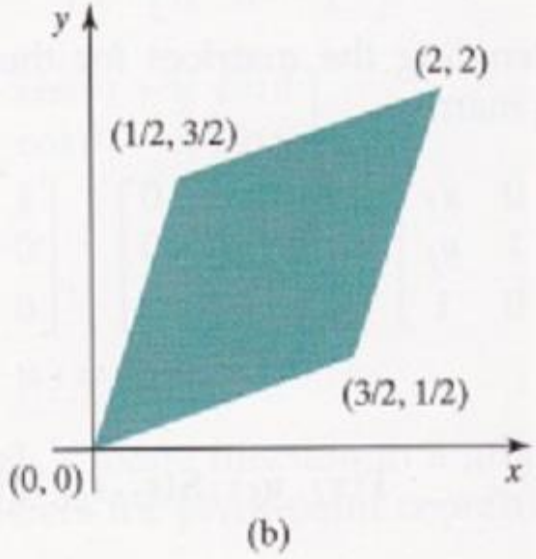
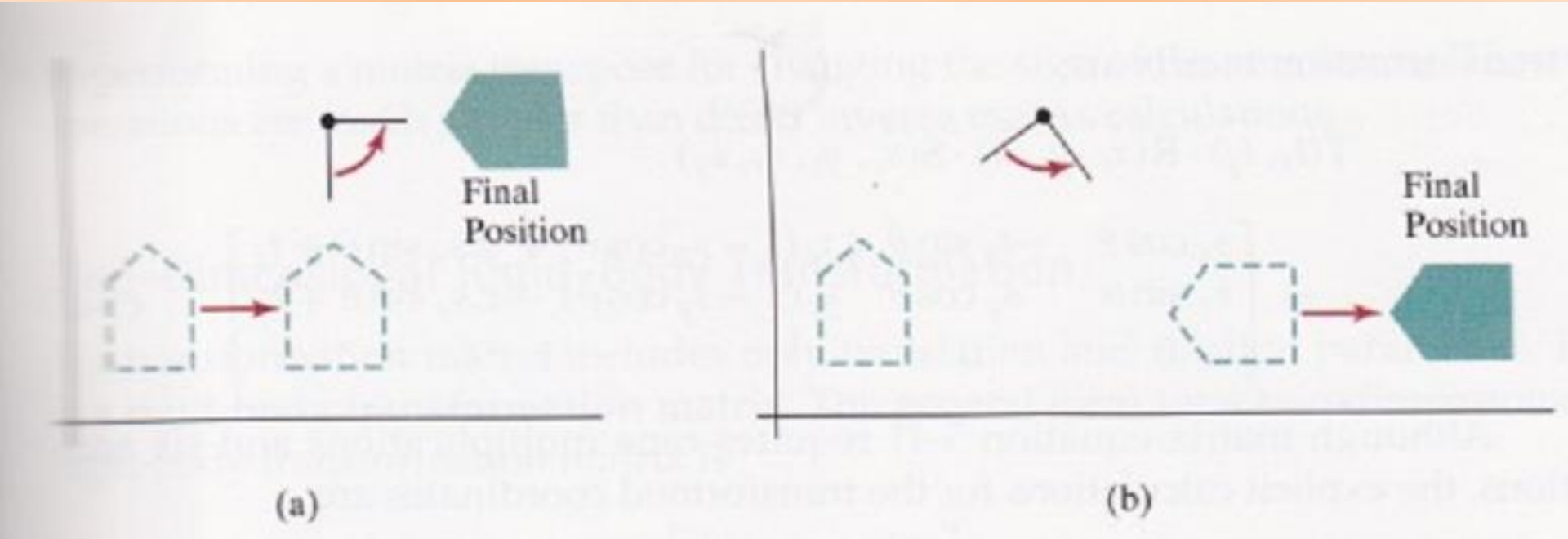


FIGURE 5-12 A square (a) is converted to a parallelogram (b) using the composite transformation matrix 5-39, with $s_1 = 1$, $s_2 = 2$, and $\theta = 45°$.

Order of Transformations

- **Changing the order in which a sequence of transformations is performed may affect the transformed position of an object**
  - **a) An object is translated then rotated .**
  - **b) An object is rotated then translated.**



2D Rigid-body Transformation

- **If a transformation matrix includes only translation and rotation parameters, it is a *rigid-body transformation matrix*.**
- **The general form for a 2D rigid-body transformation matrix is**

$$\begin{bmatrix} r_{xx} & r_{xy} & tr_x \\ r_{yx} & r_{yy} & tr_y \\ 0 & 0 & 1 \end{bmatrix}$$

Where the four elements r_{jk} are the multiplicative rotation terms, and the elements tr_x and tr_y are the translational terms.

- **A rigid-body change in coordinate position is also sometimes referred to as a rigid-motion transformation.**

2D Rigid-body Transformation

- **The matrix has the property that its upper-left 2 by 2 submatrix is an orthogonal matrix.**
- **This means that if we consider each row (or each column) of the submatrix as a vector, then the two row vectors (r_{xx}, r_{xy}) and (r_{yx}, r_{yy}) (or the two column vectors) form an orthogonal set of unit vectors. Such a set of vectors is also referred to as an orthonormal vector set. Each vector has unit length:**

$$r_{xx}^2 + r_{xy}^2 = r_{yx}^2 + r_{yy}^2 = 1$$

- **And the vectors are perpendicular (their dot product is 0):**

$$r_{xx}r_{yx} + r_{xy}r_{yy} = 0$$

2D Rigid-body Transformation

- **Therefore, if these unit vectors are transformed by the rotation submatrix, then the vector (r_{xx}, r_{xy}) is converted to a unit vector along the x axis and the vector (r_{yx}, r_{yy}) is transformed into a unit vector along the y axis of the coordinate system:**

$$\begin{bmatrix} r_{xx} & r_{xy} & 0 \\ r_{yx} & r_{yy} & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} r_{xx} \\ r_{xy} \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} r_{xx} & r_{xy} & 0 \\ r_{yx} & r_{yy} & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} r_{yx} \\ r_{yy} \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}$$

2D Rigid-body Transformation

- **The following rigid-body transformation first rotates an object through an angle θ about a pivot point (x_r, y_r) and then translates the object.**

$$T(t_x, t_y) \cdot R(x_r, y_r, \theta)$$

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

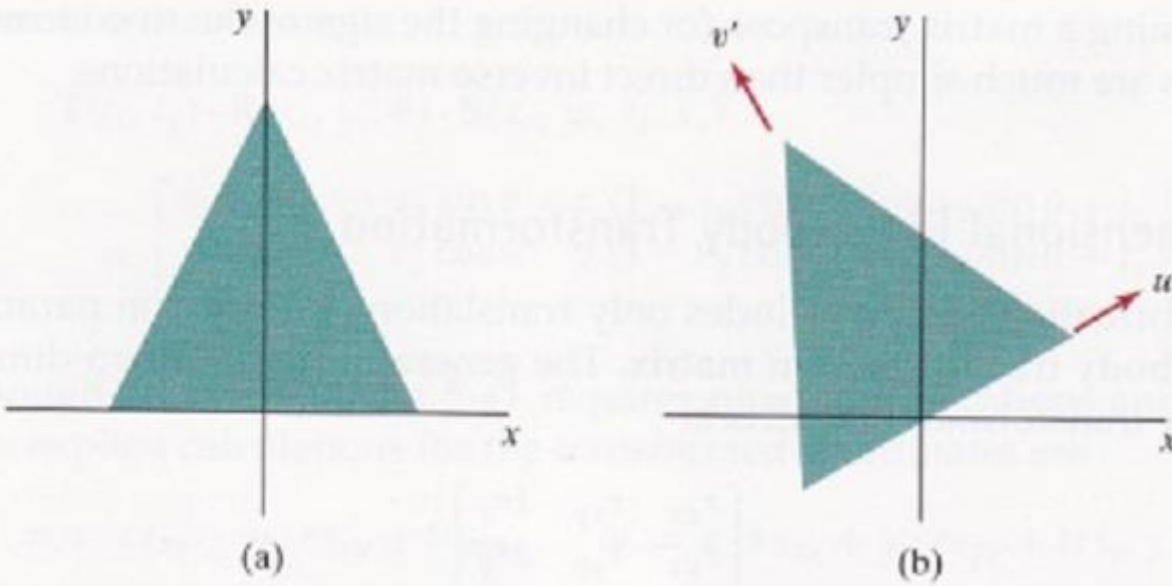


FIGURE 5-14 The rotation matrix for revolving an object from position (a) to position (b) can be constructed with the values of the unit orientation vectors **u**′ and **v**′ relative to the original orientation.

Transformations

- Last update on 14-December 2015