Higher Technological Institute
Computer Science Department

Geometric Transform

Dr Osama Farouk
Dr Ayman Soliman
Dr Adel Khaled

Outline

- **Introduction**
- **Basic Two-Dimensional Geometric Transformation**
- **Matrix Representations and Homogeneous Coordinates**

Introduction

- Operations that are applied to the geometric description of an object to change its position, orientation, or size are called geometric transformations
- Sometimes geometric-transformation operations are also referred to as modeling transformations
- A distinction between the two:

- Modeling transformations are used to construct a scene or to give the hierarchical description of a complex object that is composed of several parts
- Geometric transformations describe how objects might move around in a scene during an animation sequence or simply to view them from another angle

Basic Two-Dimensional Geometric Transformation

Available in all graphics packages are

- Translation
- Rotation
- Scaling

- Other useful transformation routines are
- Reflection
- Shearing

Two-Dimensional Translation

To translate a two-dimensional position, we add translation distances t_x and t_y to the original coordinates (x, y) to obtain the new coordinate position (x', y')

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

- Translation equations in the matrix

$$\mathbf{P} = \begin{bmatrix} x \\ y \end{bmatrix}, \qquad \mathbf{P}' = \begin{bmatrix} x' \\ y' \end{bmatrix}, \qquad \mathbf{T} = \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

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

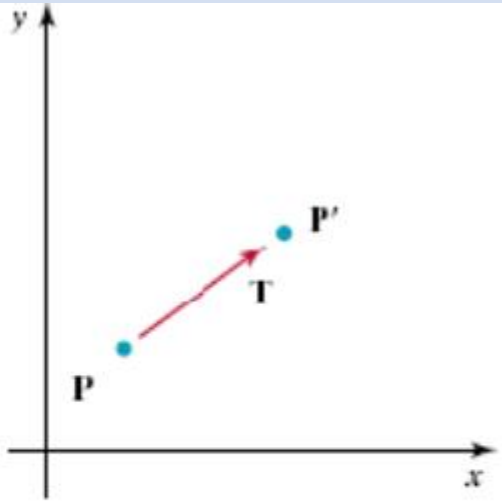

FIGURE 5-1 Translating a point from position **P** to position **P'** using a translation vector **T**.

- Translation is a rigid-body transformation that moves objects without deformation

Two-Dimensional Translation

The following routine illustrates the translation operations. An input translation vector is used to move the n vertices of a polygon from one world-coordinate position to another, and OpenGL routines are used to regenerate the translated polygon.

```
class wcPt2D {
   public:
      GLfloat x, y;
};

void translatePolygon (wcPt2D * verts, GLint nVerts, GLfloat tx, GLfloat ty)
{
   GLint k;

   for (k = 0; k < nVerts; k++) {
      verts [k].x = verts [k].x + tx;
      verts [k].y = verts [k].y + ty;
   }
   glBegin (GL_POLYGON);
      for (k = 0; k < nVerts; k++)
         glVertex2f (verts [k].x, verts [k].y);
   glEnd ( );
}
```

Polygon Translation

- A polygon is translated similarly
- Adding a translation vector to the coordinate position of each vertex and then regenerate the polygon using the new set of vertex coordinates

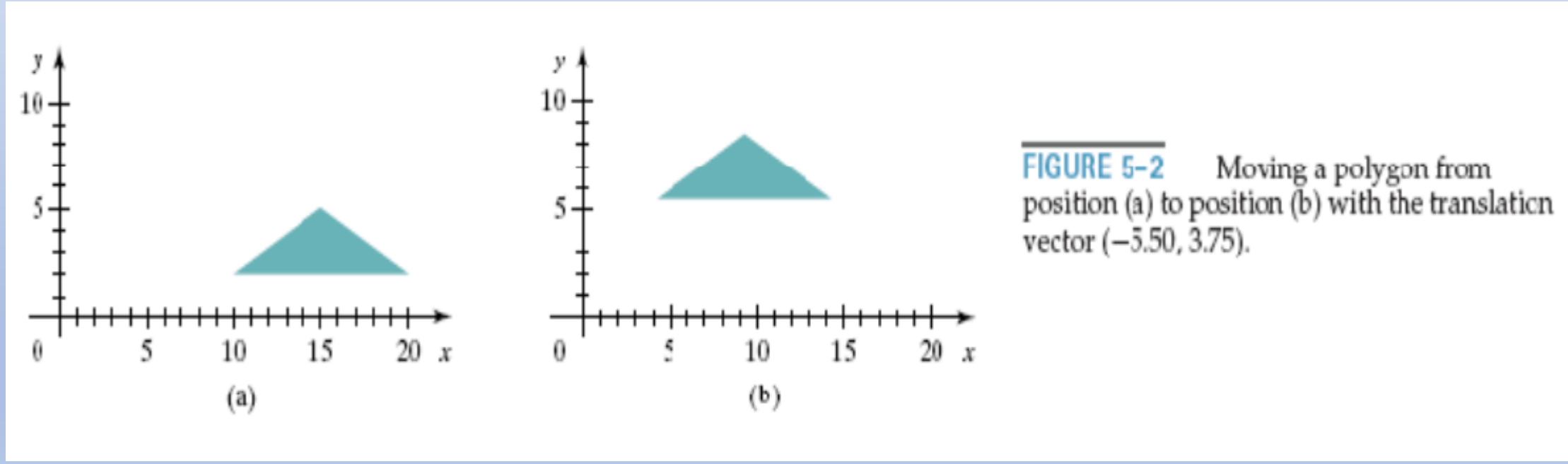

FIGURE 5-2 Moving a polygon from position (a) to position (b) with the translation vector (−5.50, 3.75).

Two-Dimensional Rotation

- A rotation transformation is generated by specifying a rotation axis and a rotation angle
- Parameters 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.
- A positive value for the angle θ defines a counterclockwise rotation about the pivot point.

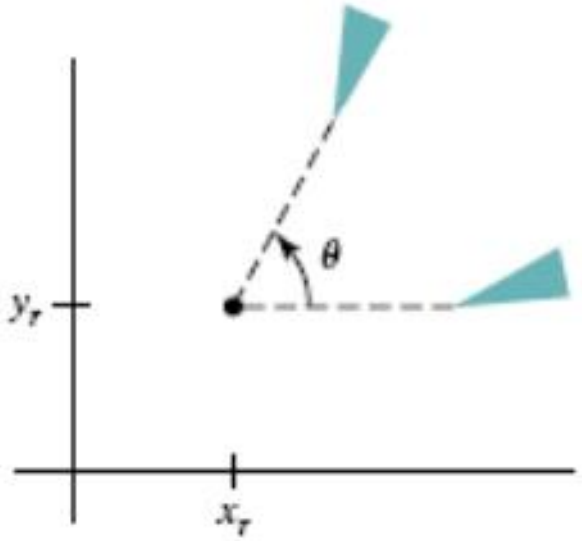

FIGURE 5-3 Rotation of an object through angle θ about the pivot point (x_r, y_r).

Two-Dimensional Rotation

- Let r is the constant distance of the point from the origin, angle ø is the original angular position of the point from the horizontal and , and θ is the rotation angle

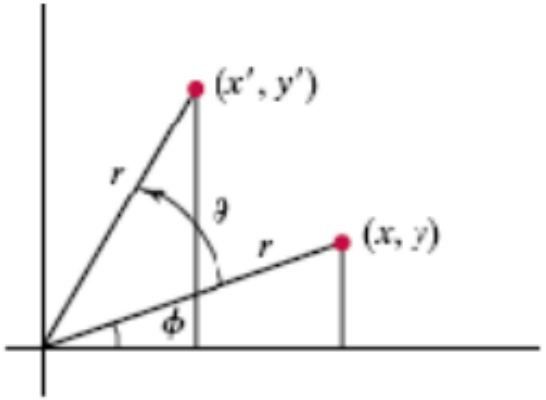

FIGURE 5–4 Rotation of a point from position (x, y) to position (x', y') through an angle θ relative to the coordinate origin. The original angular displacement of the point from the x axis is ϕ.

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

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

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

$$P' = R \cdot P$$

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

In the following code example, a polygon is rotated about a specified world coordinate pivot point. Parameters input to the rotation procedure are the original vertices of the polygon, the pivot-point coordinates, and the rotation angle **theta** specified in radians. Following the transformation of the vertex positions, the polygon is regenerated using OpenGL routines.

```
class wcPt2D {
   public:
      GLfloat x, y;
};

void rotatePolygon (wcPt2D * verts, GLint nVerts, wcPt2D pivPt,
                    GLdouble theta)
{
   wcPt2D * vertsRot;
   GLint k;

   for (k = 0; k < nVerts; k++) {
      vertsRot [k].x = pivPt.x + (verts [k].x - pivPt.x) * cos (theta)
                               - (verts [k].y - pivPt.y) * sin (theta);
      vertsRot [k].y = pivPt.y + (verts [k].x - pivPt.x) * sin (theta)
                               + (verts [k].y - pivPt.y) * cos (theta);
   }
   glBegin (GL_POLYGON);
      for (k = 0; k < nVerts; k++)
         glVertex2f (vertsRot [k].x, vertsRot [k].y);
   glEnd ( );
}
```

Rotation about Arbitrary Point

- Transformation equations for rotation of a point about any specified rotation position (x_r, y_r)

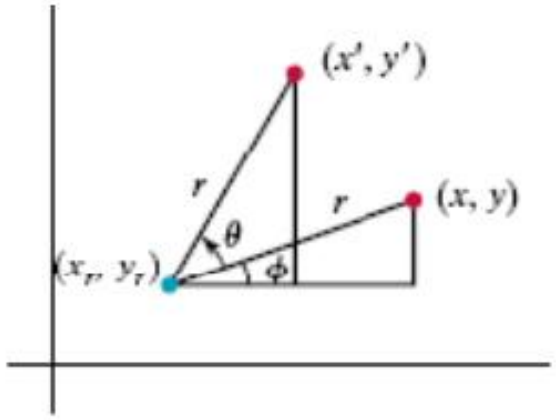

FIGURE 5-5 Rotating a point from position (x, y) to position (x', y') through an angle θ about rotation point (x_r, y_r).

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

- Rotations are rigid body transformations that move objects without deformation

Two-Dimensional Scaling

A simple two-dimensional scaling operation is performed by multiplying object positions (x, y) by scaling factors s_x and s_y to produce the transformed coordinates (x', y')

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

- In matrix format, where S is a 2x2 scaling matrix

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

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

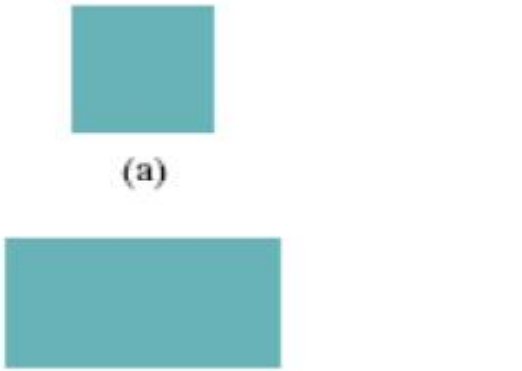

FIGURE 5–6 Turning a square (a) into a rectangle (b) with scaling factors $s_x = 2$ and $s_y = 1$.

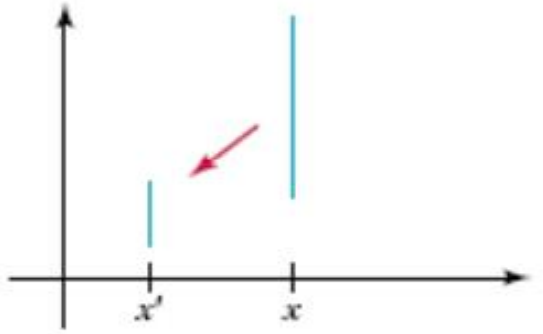

FIGURE 5–7 A line scaled with Eq. 5-12 using $s_x = s_y = 0.5$ is reduced in size and moved closer to the coordinate origin.

Scaling by a Fixed Point

- Coordinates for the fixed point, (xf, yf), are often chosen at some object position, such as its centroid
- Objects are now resized by scaling the distances between object points and the fixed point

$$x' - x_f = (x - x_f)s_x, \qquad y' - y_f = (y - y_f)s_y$$

$$x' = x \cdot s_x + x_f(1 - s_x)$$
$$y' = y \cdot s_y + y_f(1 - s_y)$$

where the additive terms $x_f(1-s_x)$ and $y_f(1-s_y)$ are constants for all points in the object

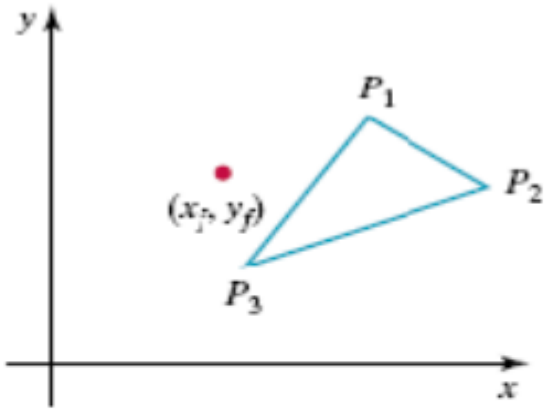

FIGURE 5-8 Scaling relative to a chosen fixed point (x_f, y_f). The distance from each polygon vertex to the fixed point is scaled by transformation equations 5-13.

The following procedure illustrates an application of the scaling calculations for a polygon. Coordinates for the polygon vertices and for the fixed point are input parameters, along with the scaling factors. After the coordinate transformations, OpenGL routines are used to generate the scaled polygon.

```
class wcPt2D {
   public:
      GLfloat x, y;
};

void scalePolygon (wcPt2D * verts, GLint nVerts, wcPt2D fixedPt,
                              GLfloat sx, GLfloat sy)
{
   wcPt2D vertsNew;
   GLint k;

   for (k = 0; k < nVerts; k++) {
      vertsNew [k].x = verts [k].x * sx + fixedPt.x * (1 - sx);
      vertsNew [k].y = verts [k].y * sy + fixedPt.y * (1 - sy);
   }
   glBegin (GL_POLYGON);
      for (k = 0; k < nVerts; k++)
         glVertex2f (vertsNew [k].x, vertsNew [k].y);
   glEnd ( );
}
```

Matrix Representations and Homogeneous Coordinates

- Many graphics applications involve sequences of geometric transformations :

- An animation might require an object to be translated and rotated at each increment of the motion

- The viewing transformations involve sequences of translations and rotations to take us from the original scene specification to the display on an output device

Matrix Representations and Homogeneous Coordinate

$$\mathbf{P}' = \mathbf{M}_1 \cdot \mathbf{P} + \mathbf{M}_2$$

- Matrix M_1 is a 2x2 array containing multiplicative factors, and M_2 is a two-element column matrix containing translational terms
- Multiplicative and translational terms can be combined into a single matrix if we expand the representations to 3x3
- A three-element representation $(x_h,\ y_h,\ h)$, called homogeneous coordinates, where the homogeneous parameter h is a nonzero value such that

$$x = \frac{x_h}{h}, \qquad y = \frac{y_h}{h}$$

Matrix Representations and Homogeneous Coordinate

- Two-Dimensional Translation Matrix

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

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

- Rotation Matrix

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

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

- Scaling Matrix

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

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