Higher Technological Institute
Computer Science Department

Geometric Transform
Lec. 07

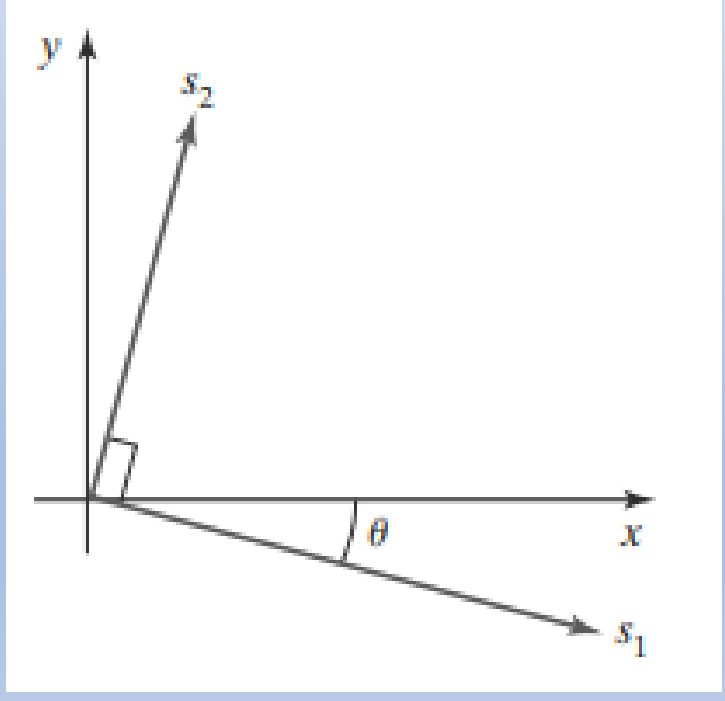
Dr Osama Farouk
Dr Ayman Soliman
Dr Adel Khaled

Outlines

- General Two-Dimensional Scaling Directions.
- Matrix Concatenation Properties.
- General Two-Dimensional Composite Transformations and Computational Efficiency.
- Two-Dimensional Rigid-Body Transformation.

General Two-Dimensional 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.



Scaling with rotation

$$\cdot \mathbf{S}(s_1, s_2) \cdot \mathbf{R}(\theta)$$

General Two-Dimensional Scaling Directions

To accomplish the scaling without changing the orientation of the object,

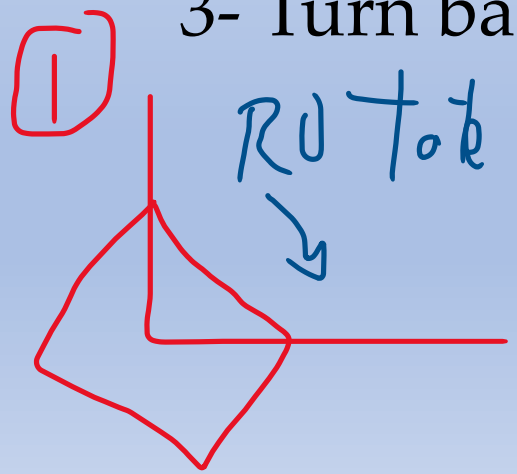
- we first perform a rotation so that the directions for $s1$ and $s2$ coincide with the x and y axes, respectively.
- Then the scaling transformation $\mathbf{S}(s1, s2)$ is applied, followed by an opposite rotation to return points to their original orientations.
- The composite matrix resulting from the product of these three transformations is

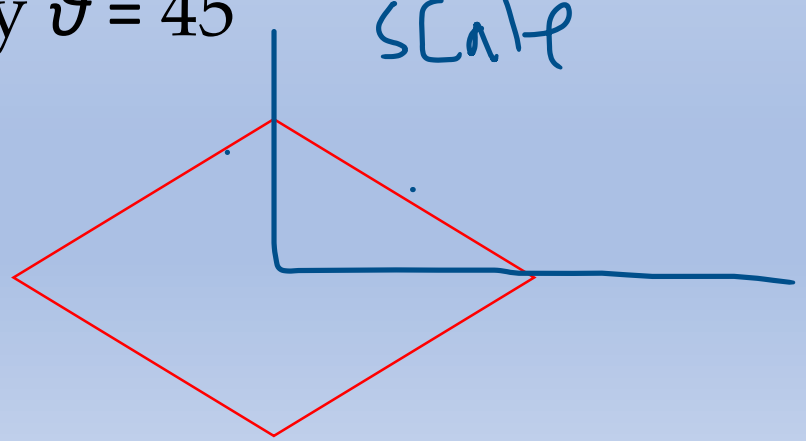
$$\mathbf{R}^{-1}(\theta) \cdot \mathbf{S}(s_1, s_2) \cdot \mathbf{R}(\theta) = \begin{bmatrix} s_1 \cos^2\theta + s_2 \sin^2\theta & (s_2 - s_1)\cos\theta \sin\theta & 0 \\ (s_2 - s_1)\cos\theta \sin\theta & s_1 \sin^2\theta + s_2 \cos^2\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

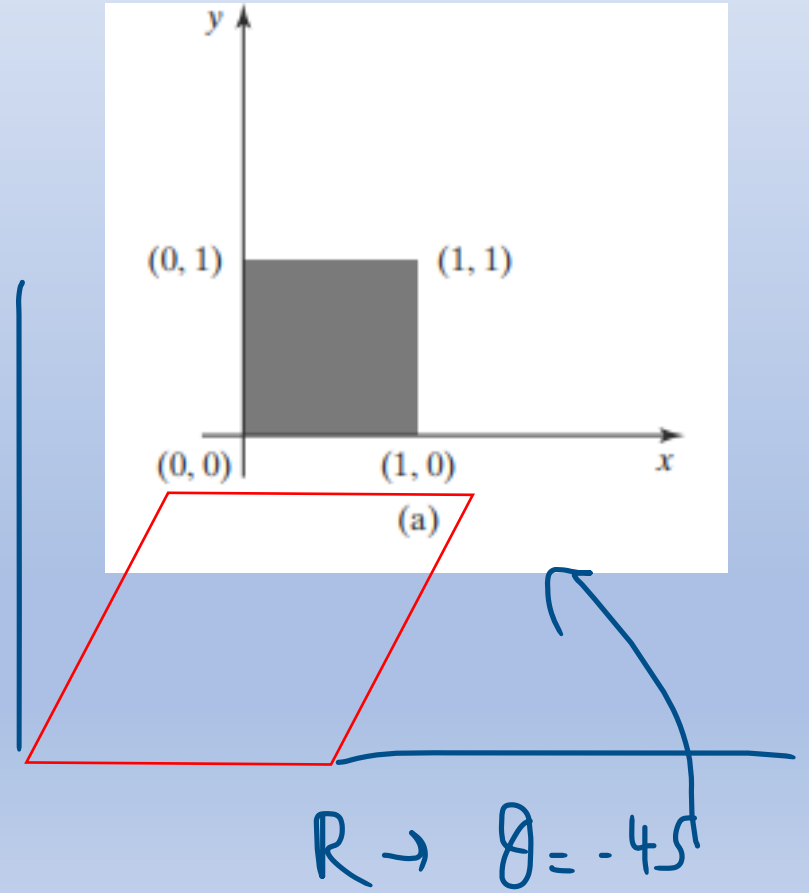
General Two-Dimensional Scaling Directions

Example

1- Rotate by $\vartheta = 45$
2-Stretch the square shown in figure a, in x axis direction by factor 1 and in y-direction by factor 2, $s_x = 1$ and $s_y = 2$.
3- Turn back by $\vartheta = 45$

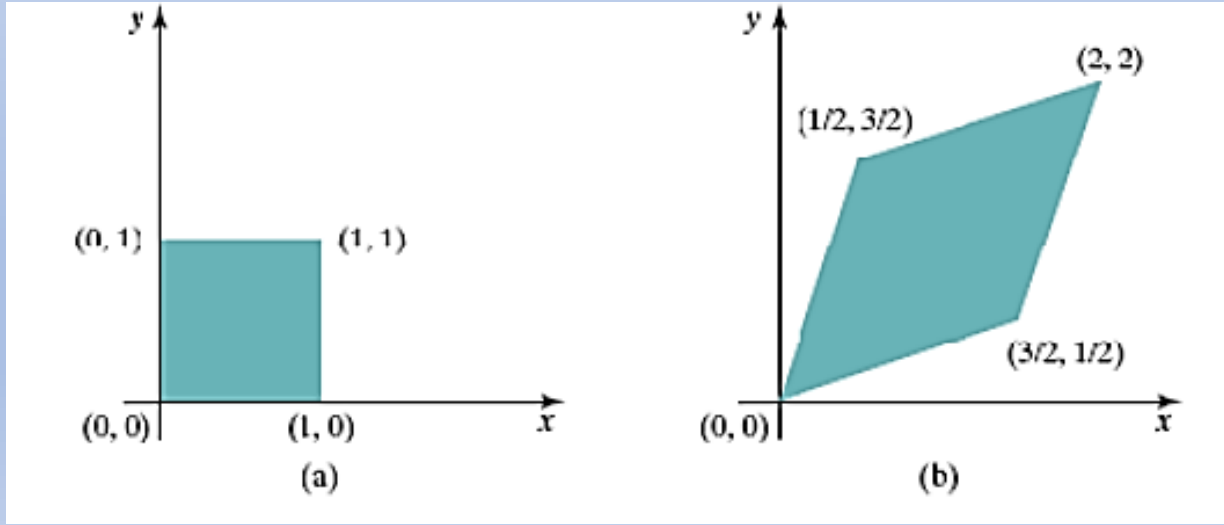






General Two-Dimensional Scaling Directions

- Turn a unit square into a parallelogram by stretching it along the diagonal from (0, 0) to (1, 1)
  - Rotate the diagonal onto the y axis using $\theta = 45^\circ$
  - Double its length with the scaling values $s_1=1$ and $s_2=2$
  - Rotate again to return the diagonal to its original orientation



Matrix Concatenation Properties

Depending upon the order in which the transformations are specified, we can construct a composite matrix either by multiplying from left to right (premultiplying) or by multiplying from right to left (postmultiplying).

- Multiplication of matrices is associative

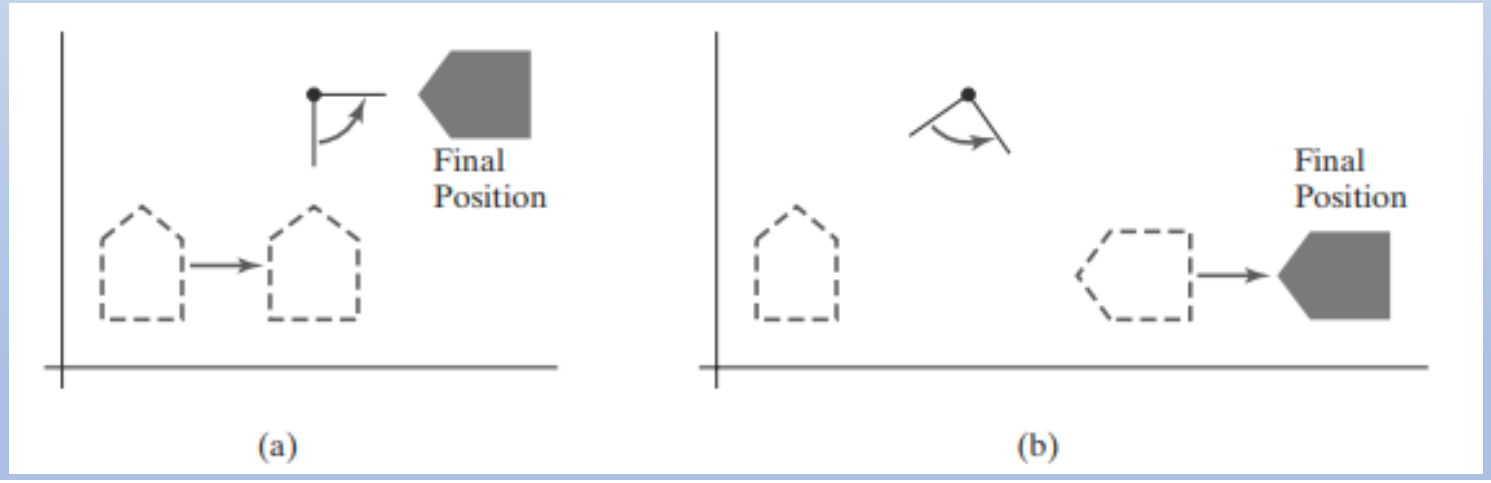
$$M_3 \cdot M_2 \cdot M_1 = (M_3 \cdot M_2) \cdot M_1 = M_3 \cdot (M_2 \cdot M_1)$$

- Depending upon the order in which the transformations
  - by multiplying from left-to-right (premultiplying)
  - by multiplying from right-to-left (postmultiplying)

  - In OpenGL's convention:
    - V'=B*(A*V), V'=(B*A)*V, C=B*A; V'=C*V;
  - In DX's convention:
    - V'=(V*A)*B, C=A*B; V'=V*C;

Matrix Concatenation Properties

- Transformation products, on the other hand, may not be commutative. تبادلي
  - M2 × M1 is not equal to M1 × M2
- This means that if we want to translate and rotate an object, we must be careful about the order in which the composite matrix is evaluated



Reversing the order in which a sequence of transformations is performed may affect the transformed position of an object. In (a), an object is first translated in the *x* direction, then rotated counterclockwise through an angle of 45°. In (b), the object is first rotated 45° counterclockwise, then translated in the *x* direction.

General Two-Dimensional Composite Transformations and Computational Efficiency

A two-dimensional transformation, representing any combination of translations, rotations, and scaling, can be expressed as

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} rs_{xx} & rs_{xy} & trs_x \\ rs_{yx} & rs_{yy} & trs_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Eqn(1)

The four elements rs_{jk} are the multiplicative rotation-scaling terms in the transformation, which involve only rotation angles and scaling factors. Elements trs_x and trs_y are the translational terms

General Two-Dimensional Composite Transformations and Computational Efficiency

For example, if an object is to be scaled and rotated about its centroid coordinates (x_c, y_c) and then translated, the values for the elements of the composite transformation matrix are

$$\mathbf{T}(t_x, t_y) \cdot \mathbf{R}(x_c, y_c, \theta) \cdot \mathbf{S}(x_c, y_c, s_x, s_y)$$

$$= \begin{bmatrix} s_x \cos\theta & -s_y \sin\theta & x_c(1 - s_x \cos\theta) + y_c s_y \sin\theta + t_x \\ s_x \sin\theta & s_y \cos\theta & y_c(1 - s_y \cos\theta) - x_c s_x \sin\theta + t_y \\ 0 & 0 & 1 \end{bmatrix}$$

Eqn(2)

- From eqn(1) it requires , it requires 9 multiplications and 6 additions
- Actually, only 4 multiplications and 4 additions

$$x' = x \cdot rs_{xx} + y \cdot rs_{xy} + trs_x, \qquad y' = x \cdot rs_{yx} + y \cdot rs_{yy} + trs_y$$

General Two-Dimensional Composite Transformations and Computational Efficiency

- Once matrix is concatenated, it is maximum number of computations required.
- Without concatenation, individual transformations would be applied one at a time, and the number of calculations could be significantly increase.

Computational Efficiency

- Rotation calculations require trigonometric evaluations and several multiplications
- Computational efficiency can become an important consideration in rotation transformations
- For small enough angles (less than 10°), cosθ is approximately 1.0 and sinθ has a value very close to the value of θ in radians

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

$$x' = x - y\sin\theta, \qquad y' = x\sin\theta + y$$

Computational Efficiency

- Composite transformations often involve inverse matrices
- Operations are much simpler than direct inverse matrix calculations
  - Inverse translation matrix is obtained by changing the signs of the translation distances
  - Inverse rotation matrix is obtained by performing a matrix transpose

Two-Dimensional Rigid-Body Transformation

If a transformation matrix includes only translation and rotation parameters, it is a **rigid-body transformation matrix**

- All angles and distances between coordinate positions are unchanged by the transformation

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

- Upper-left 2x2 submatrix is an orthogonal matrix
  - Two row vectors (r_{xx}, r_{xy}) and (r_{yx}, r_{yy}) (or the two column vectors) form an orthogonal set of unit vectors
  - Set of vectors is also referred to as an orthonormal vector set

Two-Dimensional Rigid-Body Transformation

- Each vector has unit length

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

- Their dot product is 0

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

- If these unit vectors are transformed by the rotation sub-matrix, then

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

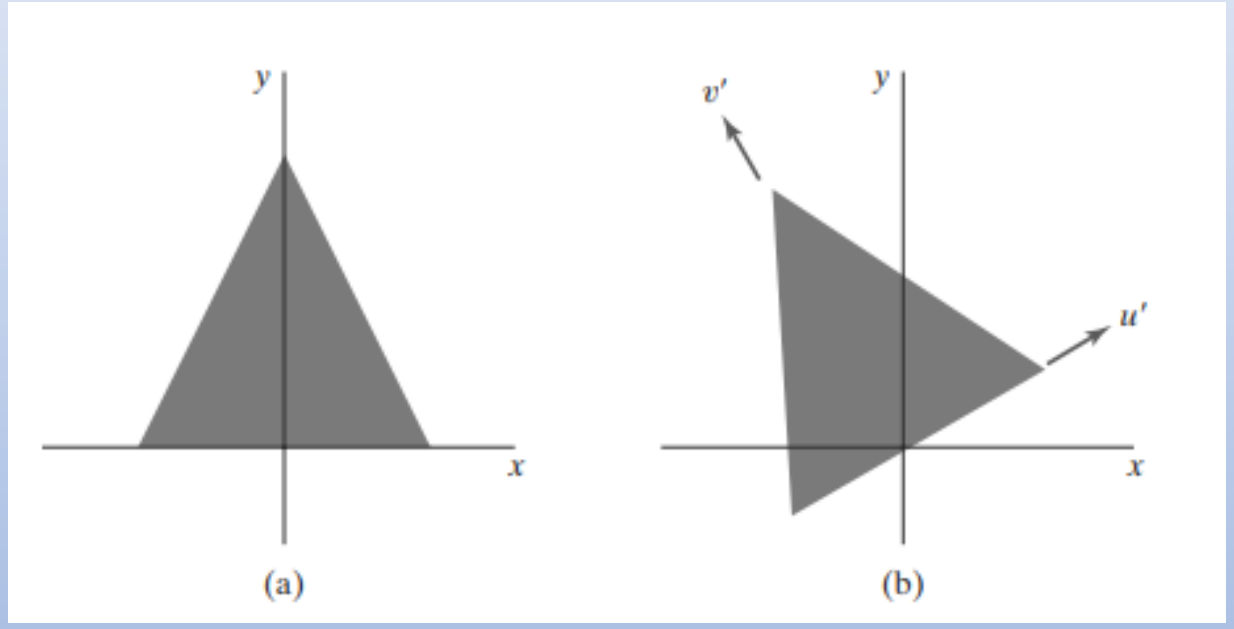
$$\begin{bmatrix} r_{xx} & r_{xy} & 0 \\ r_{yx} & r_{yy} & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} r_{yx} \\ r_{yy} \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}$$

- Example,

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

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

Two-Dimensional Rigid-Body Transformation



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.

End of Lecture
Good Luck!

See you

in next lecture...