Computer Graphics

Lecture 5

Transformations

Geometric Transformations as Matrices

- Operations to arrange objects in a scene, view them with cameras, and get them on a screen all can be encoded with linear algebra
  - Geometric operations: rotation, translation, scaling, projection, and more…
- These transforms operate differently on points, displacement vectors, and surface normals

2D Linear Transformations

- We can transform points in a 2D coordinate system by multiplying the point (a vector) by a matrix (the transformation):

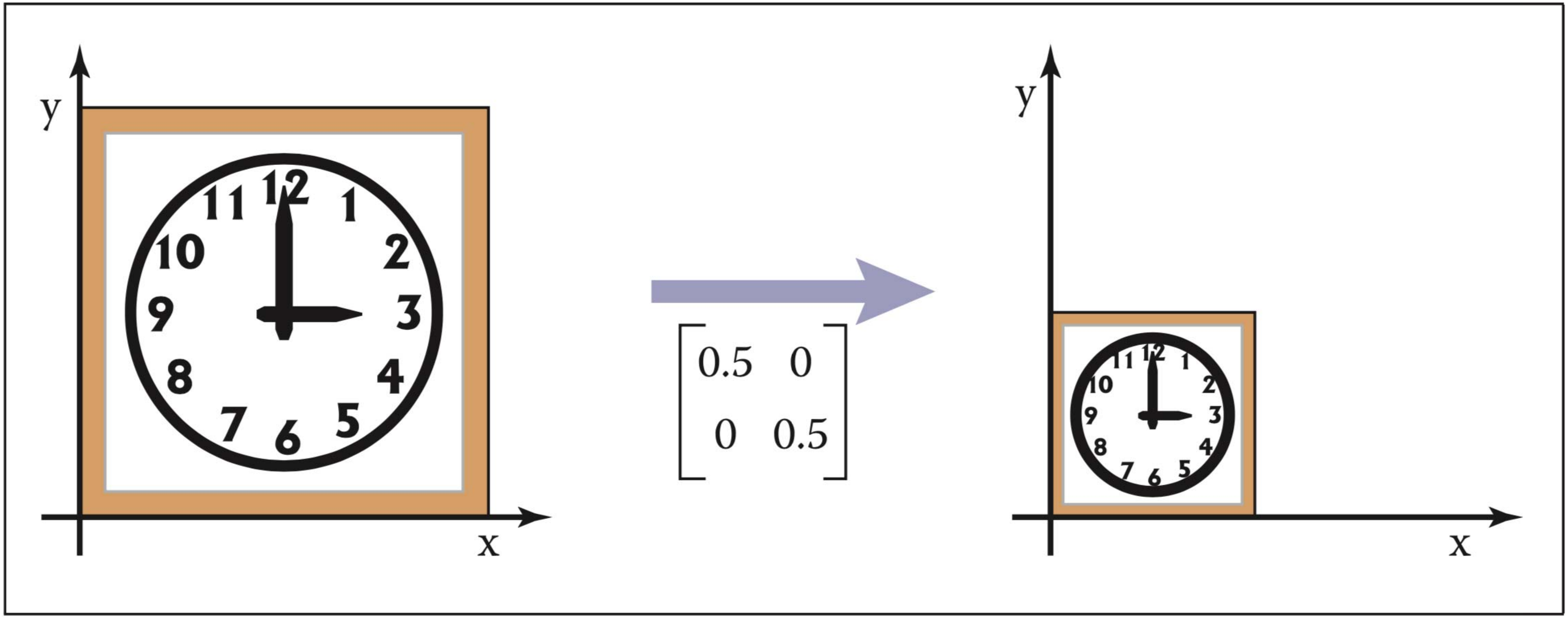
- e.g. Multiply the matrix A by **x** = (x,y), or A**x**:

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a_{11}x + a_{12}y \\ a_{21}x + a_{22}y \end{bmatrix}$$

- Such transformations are called *linear* because they satisfy the relationship that $A(a\mathbf{x}_1+\mathbf{x}_2) = aA\mathbf{x}_1 + A\mathbf{x}_2$

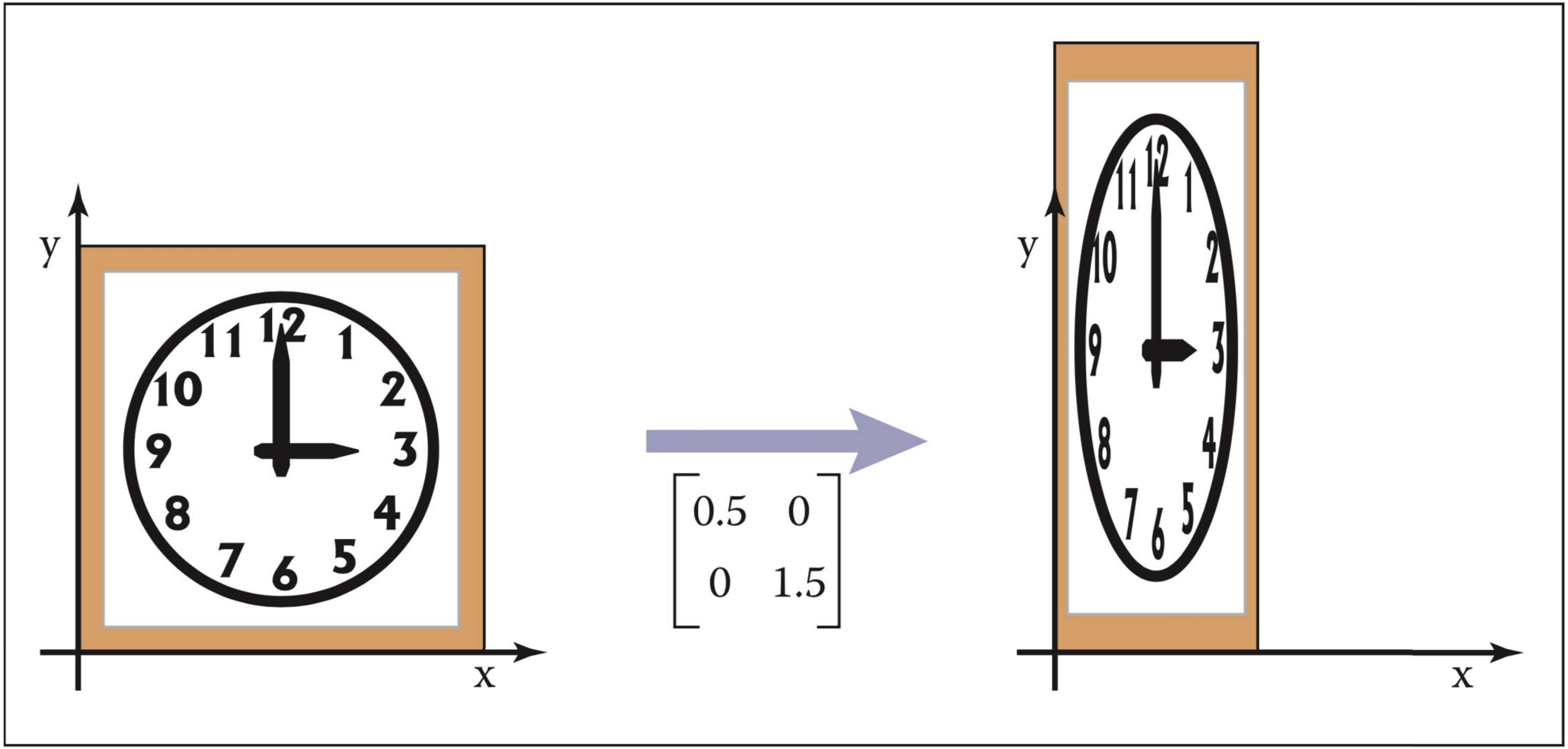
Scaling

- Can be uniform, where $s_x = s_y$. $$\begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} s_x x \\ s_y y \end{bmatrix}$$



Scaling

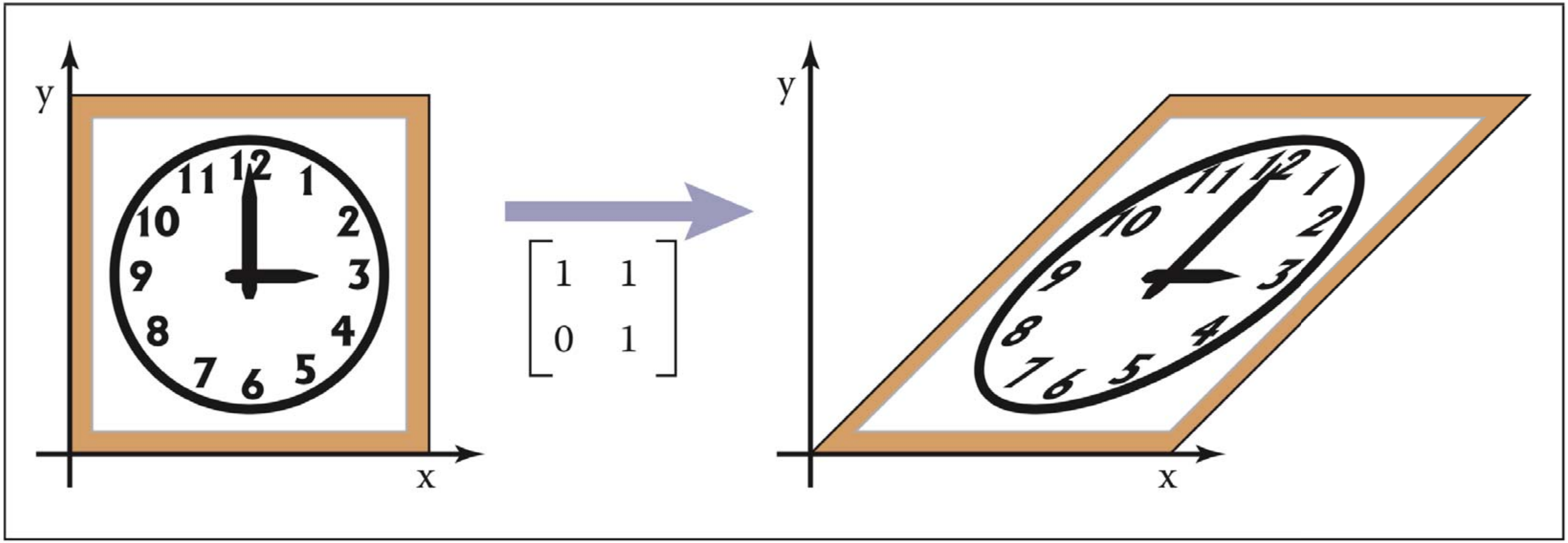
- Can also be nonuniform, where $s_x \neq s_y$.



Shearing

- Horizontal shearing shifts each row based on the y value.

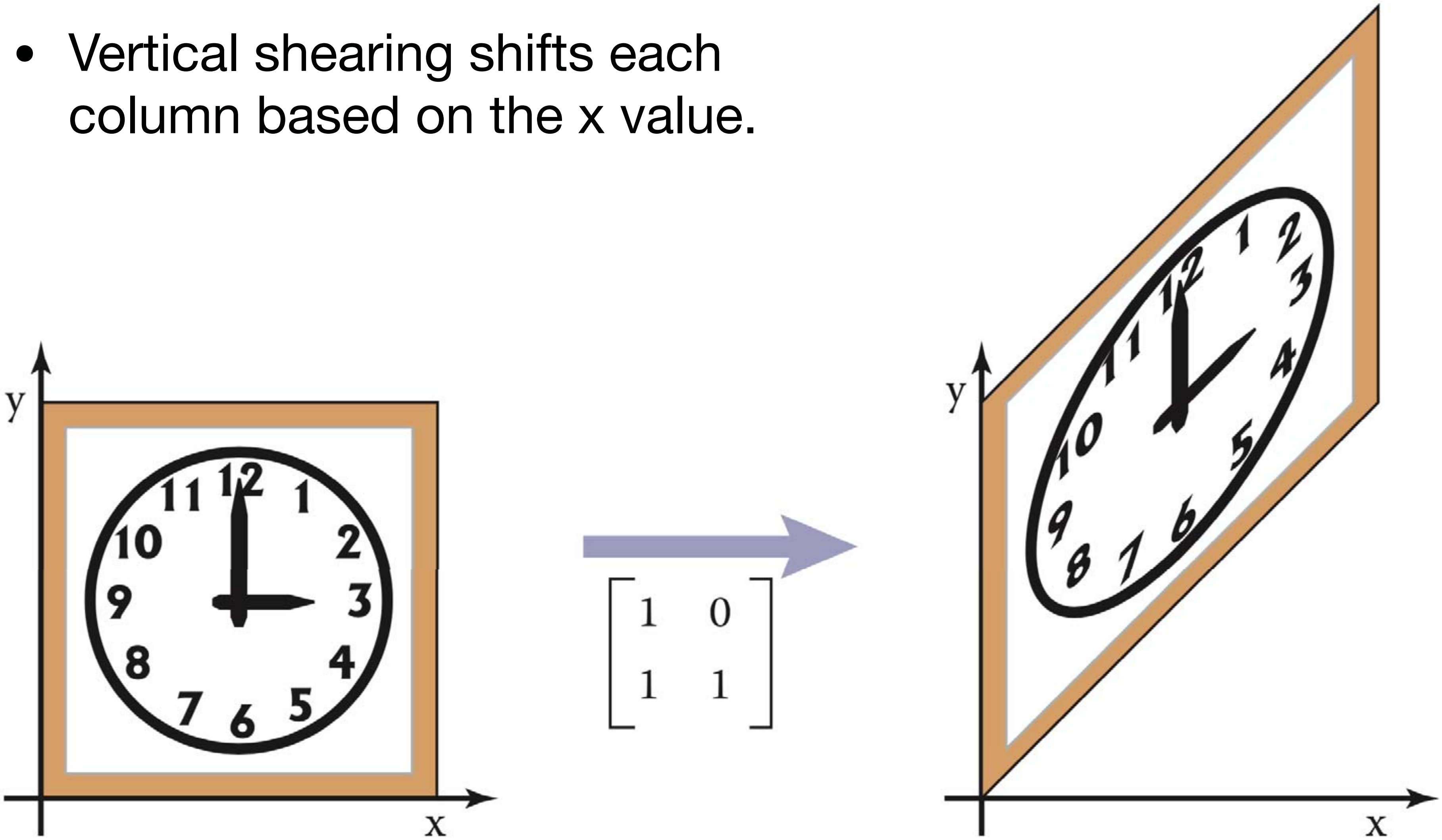
$$\text{shear-x}(s) = \begin{bmatrix} 1 & s \\ 0 & 1 \end{bmatrix}$$



Shearing

$$\text{shear-y}(s) = \begin{bmatrix} 1 & 0 \\ s & 1 \end{bmatrix}$$

- Vertical shearing shifts each column based on the x value.

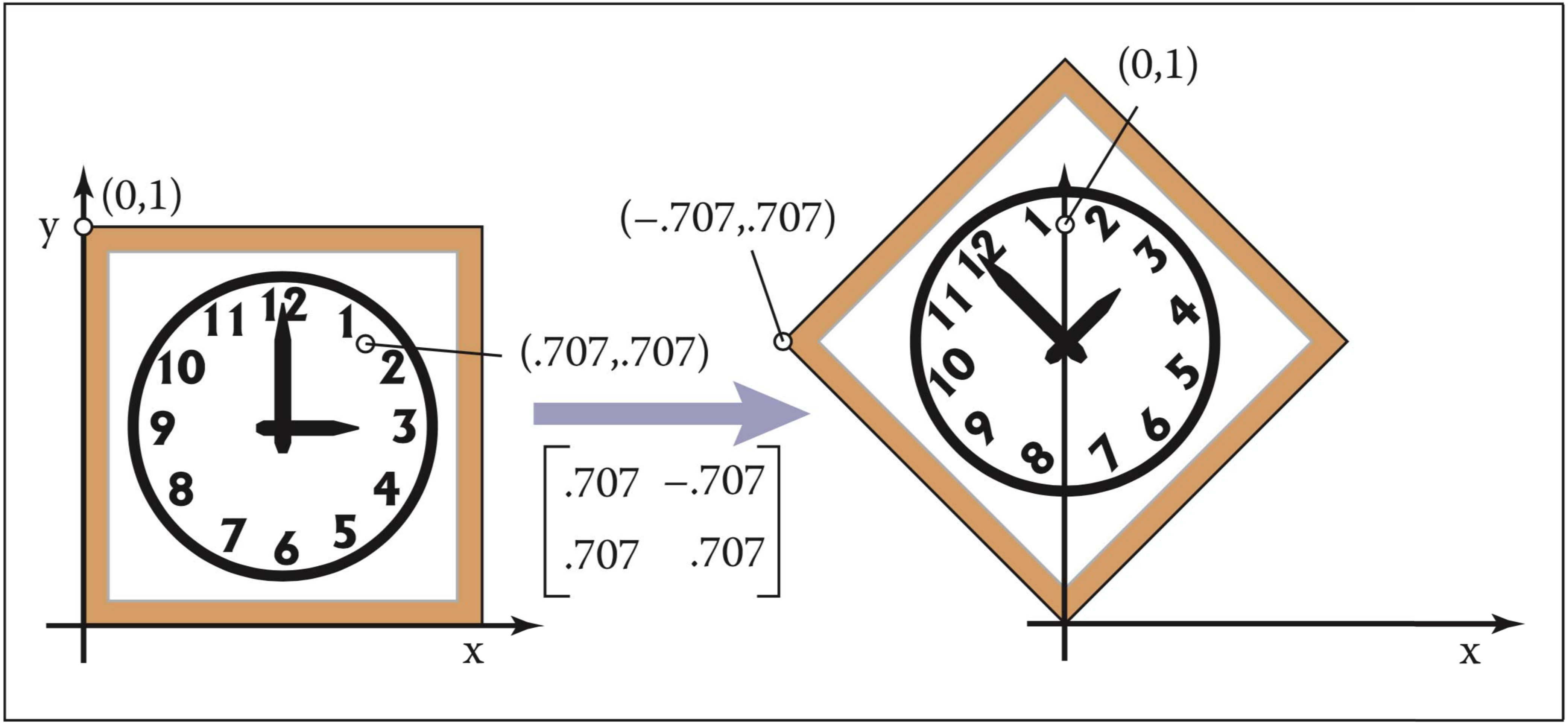


Rotation

$$\text{rotate}(\phi) = \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix}$$

- Rotate counterclockwise by an angle ϕ about the origin.

$$\begin{bmatrix} \cos\frac{\pi}{4} & -\sin\frac{\pi}{4} \\ \sin\frac{\pi}{4} & \cos\frac{\pi}{4} \end{bmatrix} = \begin{bmatrix} 0.707 & -0.707 \\ 0.707 & 0.707 \end{bmatrix}$$

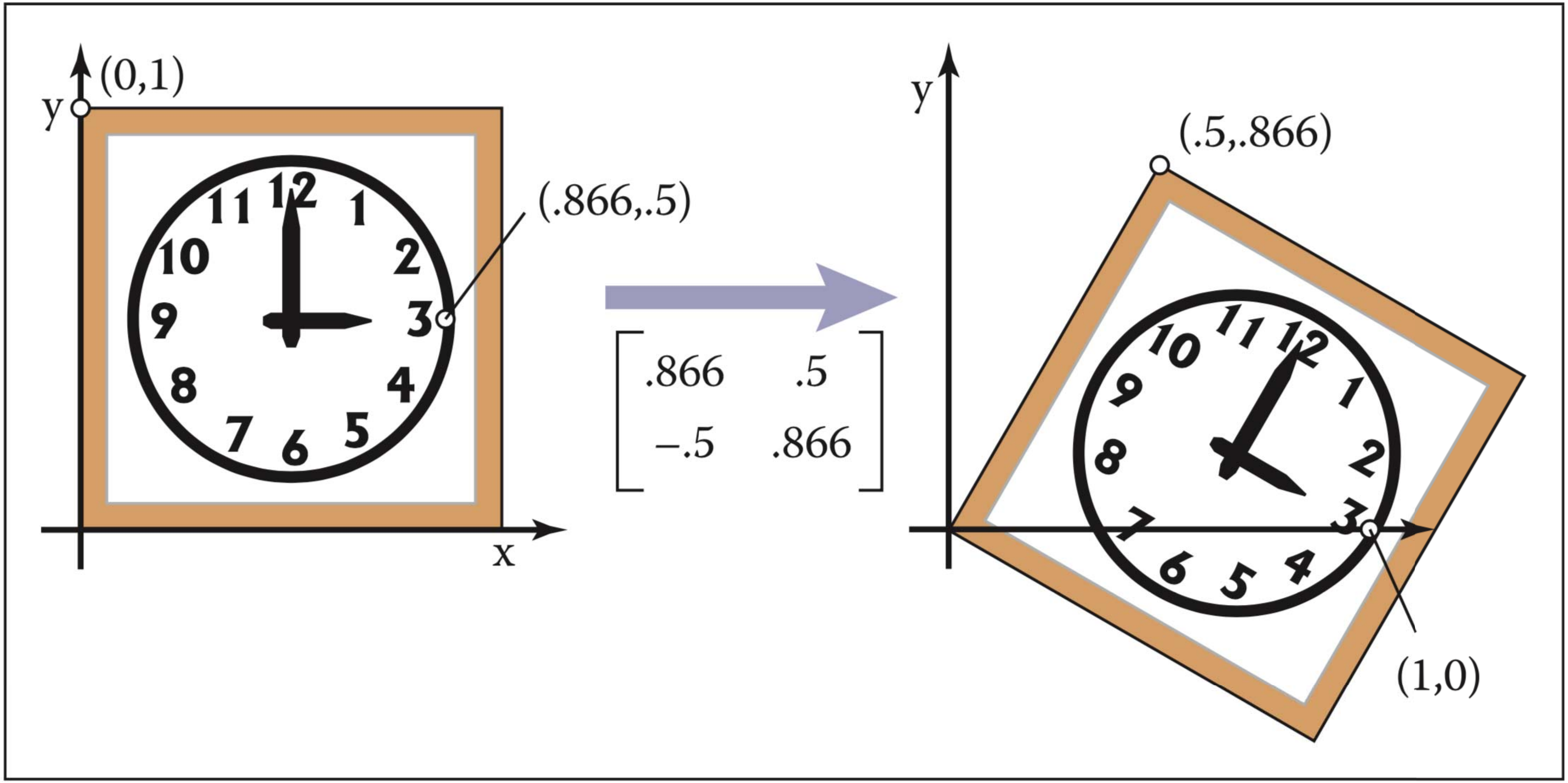


Rotation

$$\text{rotate}(\phi) = \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix}$$

- Clockwise rotations can also be represented by negative angles

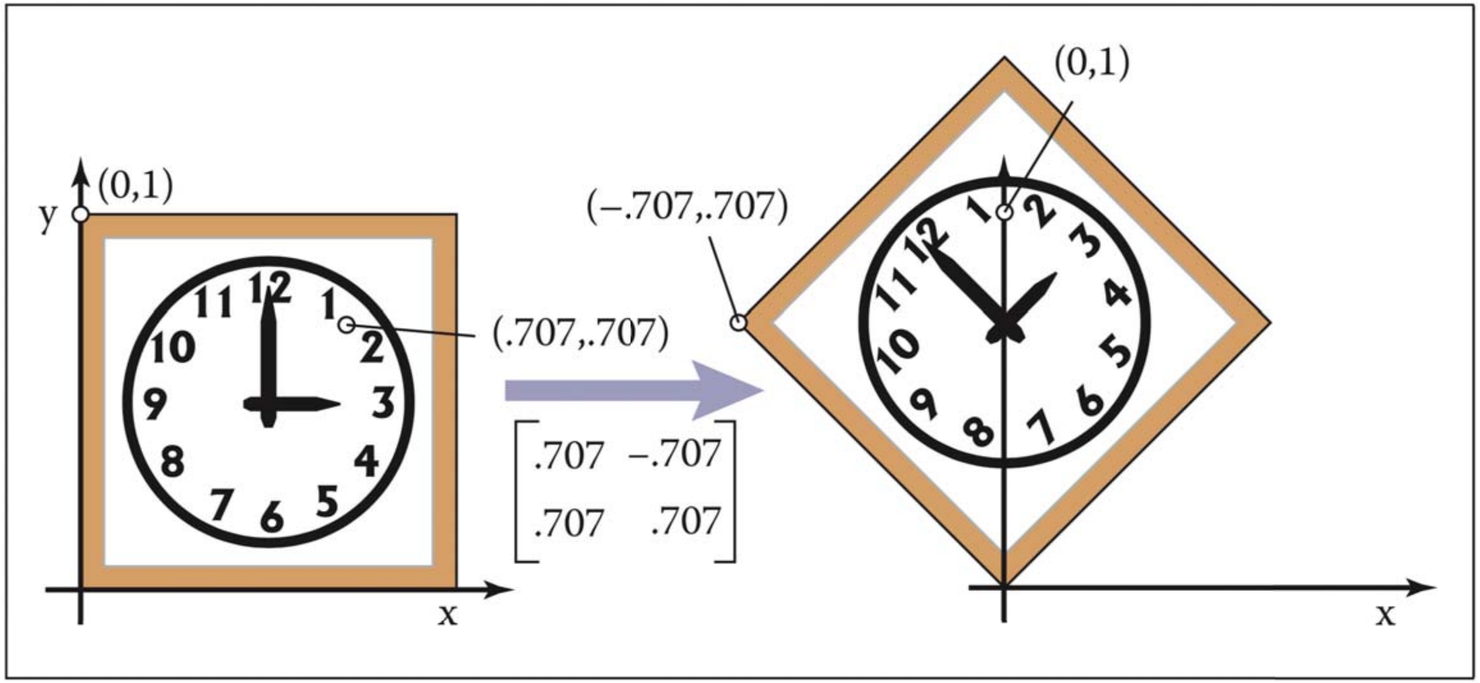
$$\begin{bmatrix} \cos\frac{-\pi}{6} & -\sin\frac{-\pi}{6} \\ \sin\frac{-\pi}{6} & \cos\frac{-\pi}{6} \end{bmatrix} = \begin{bmatrix} 0.866 & 0.5 \\ -0.5 & 0.866 \end{bmatrix}$$



Rotations are Always Orthogonal Matrices

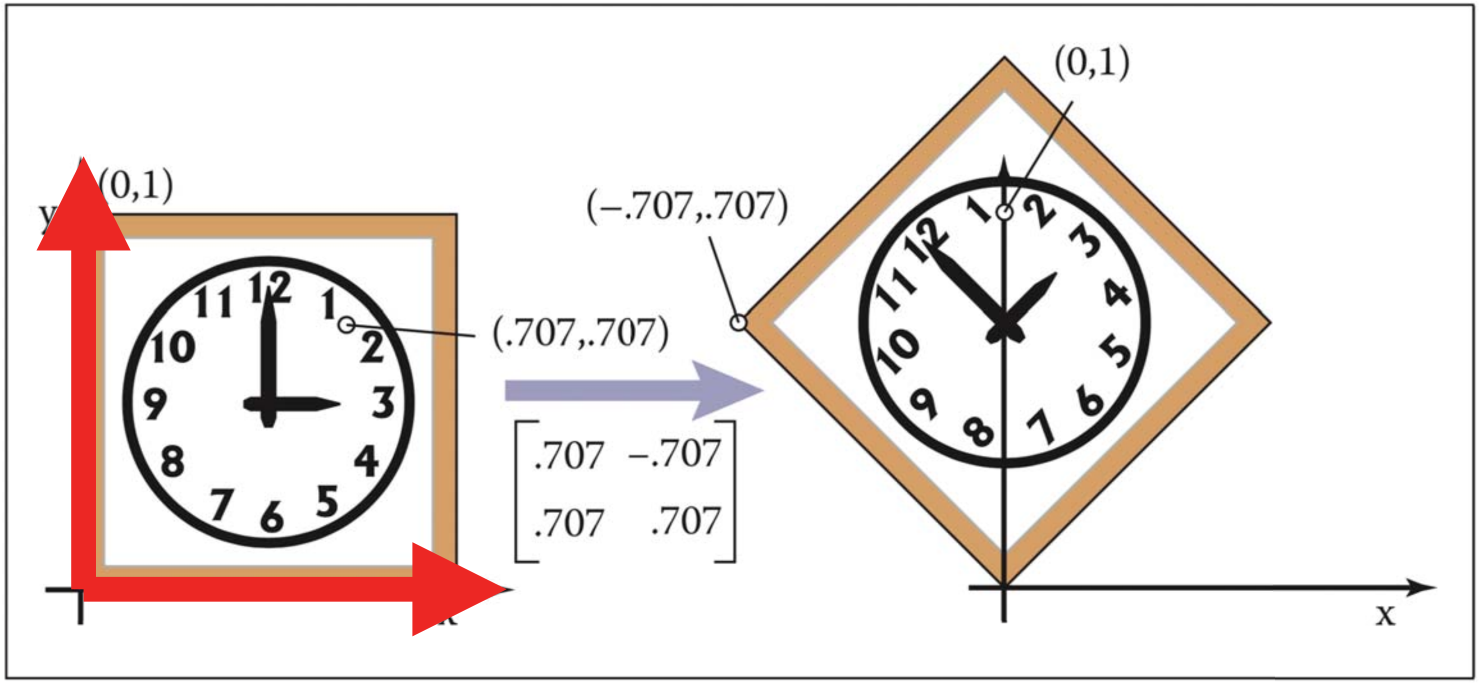
- Recall: An **orthogonal matrix** always has columns and rows that are orthogonal unit vectors

- Implication: Have the effect of rotating the coordinate axes



Rotations are Always Orthogonal Matrices

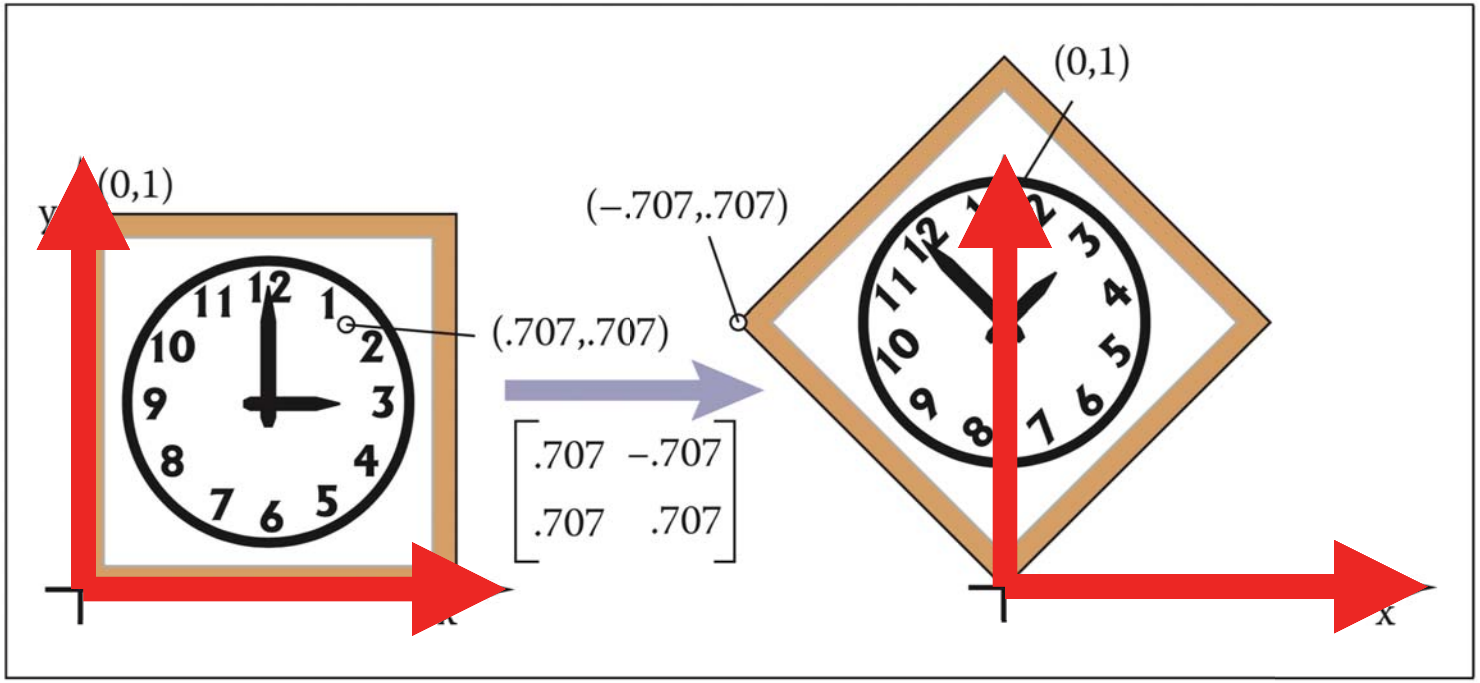
- Recall: An **orthogonal matrix** always has columns and rows that are orthogonal unit vectors
- Implication: Have the effect of rotating the coordinate axes



Rotations are Always Orthogonal Matrices

- Recall: An **orthogonal matrix** always has columns and rows that are orthogonal unit vectors

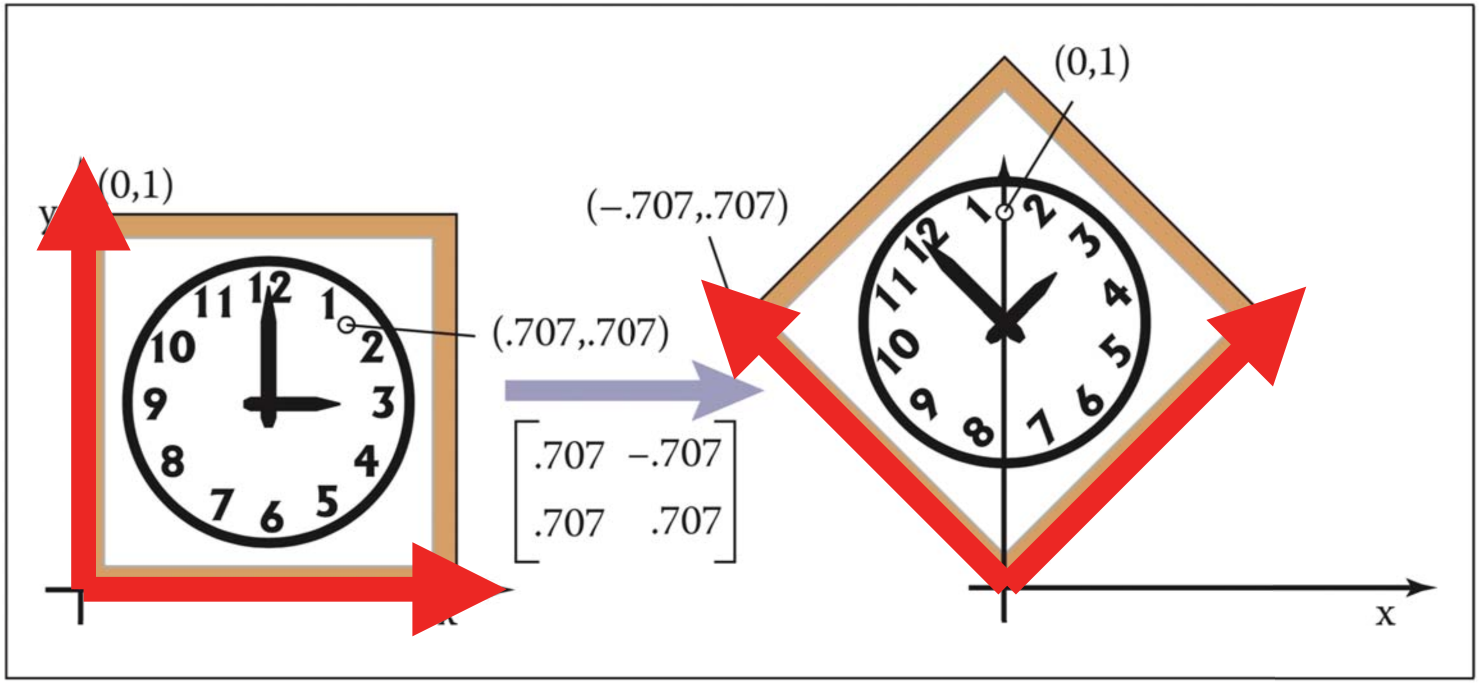
- Implication: Have the effect of rotating the coordinate axes



Rotations are Always Orthogonal Matrices

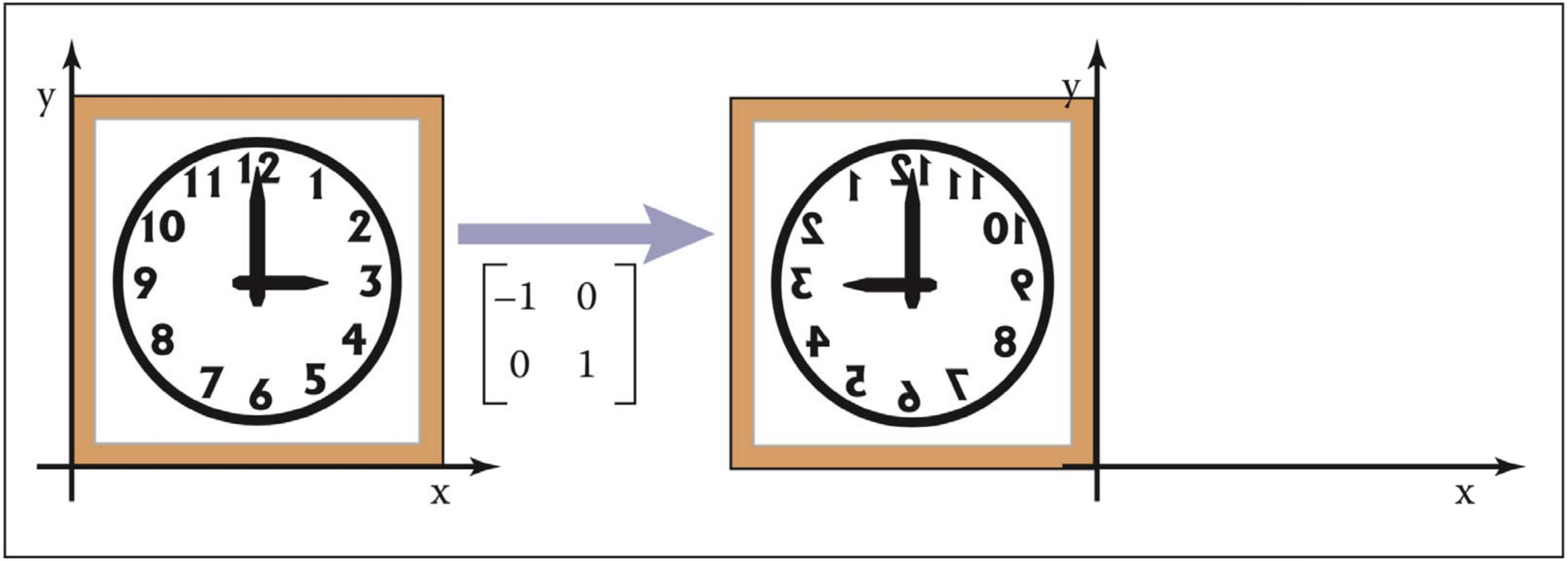
- Recall: An **orthogonal matrix** always has columns and rows that are orthogonal unit vectors

- Implication: Have the effect of rotating the coordinate axes

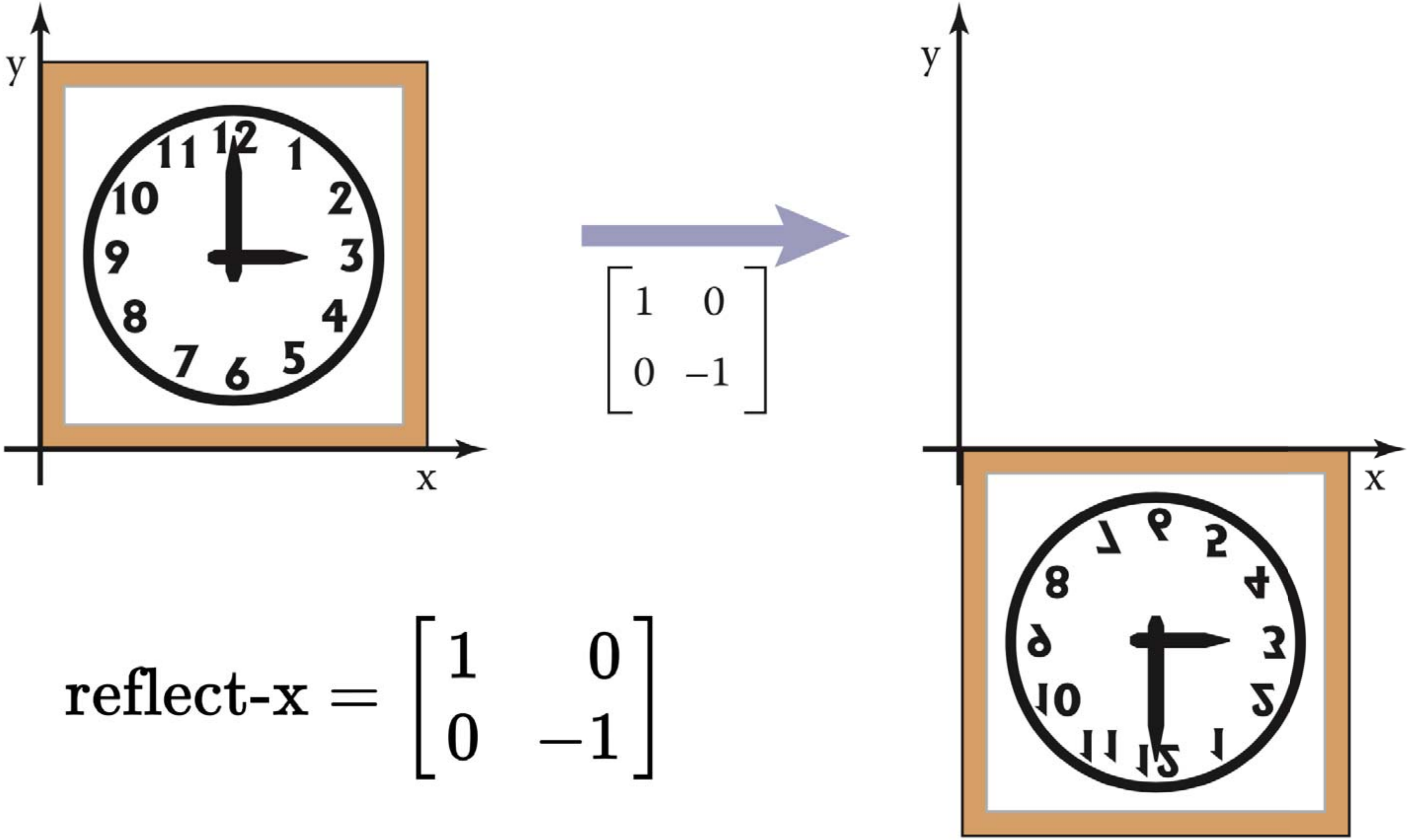


Reflection

$$\text{reflect-y} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}$$



Reflection



$$\text{reflect-x} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

Composition

- Transformations can be **composed** to perform combinations of transformations.

- For example, one could first rotate with matrix R and then scale with matrix S

- Applied to a point $\mathbf{v_1}$, this would be

    - $\mathbf{v_2} = R\mathbf{v_1}$ to rotate and then

    - $\mathbf{v_3} = S\mathbf{v_2} = SR\mathbf{v_1}$
      $= (SR)\mathbf{v_1} = T\mathbf{v_1}$ where $T = SR$

Order Matters for Composition

- First rotate, then non-uniform scale
- Rotate:

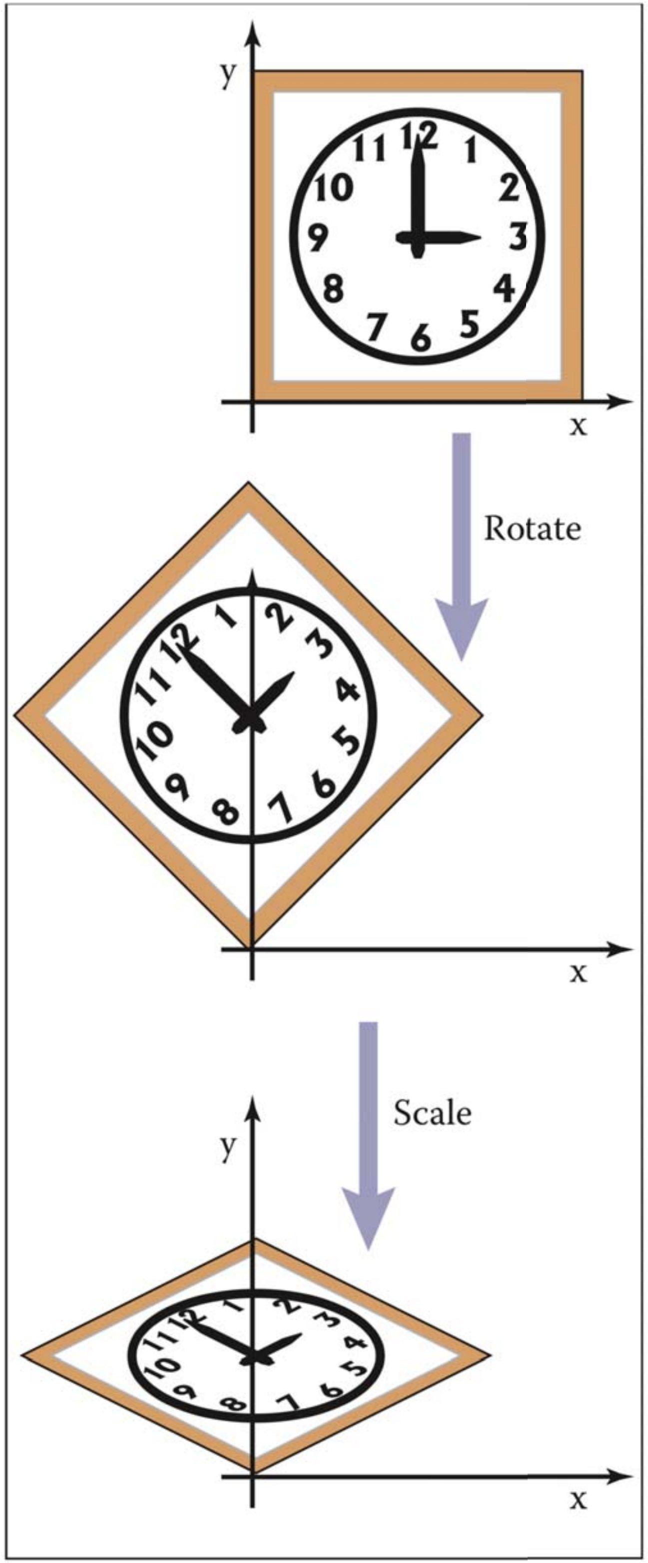
$$\begin{bmatrix} 0.707 & -0.707 \\ 0.707 & 0.707 \end{bmatrix}$$

- Scale:

$$\begin{bmatrix} 1 & 0 \\ 0 & 0.5 \end{bmatrix}$$

- Combined:

$$\begin{bmatrix} 1 & 0 \\ 0 & 0.5 \end{bmatrix} \begin{bmatrix} 0.707 & -0.707 \\ 0.707 & 0.707 \end{bmatrix} = \begin{bmatrix} 0.707 & -0.707 \\ 0.353 & 0.353 \end{bmatrix}$$



Order Matters for Composition

- First non-uniform scale, then rotate

- Rotate:

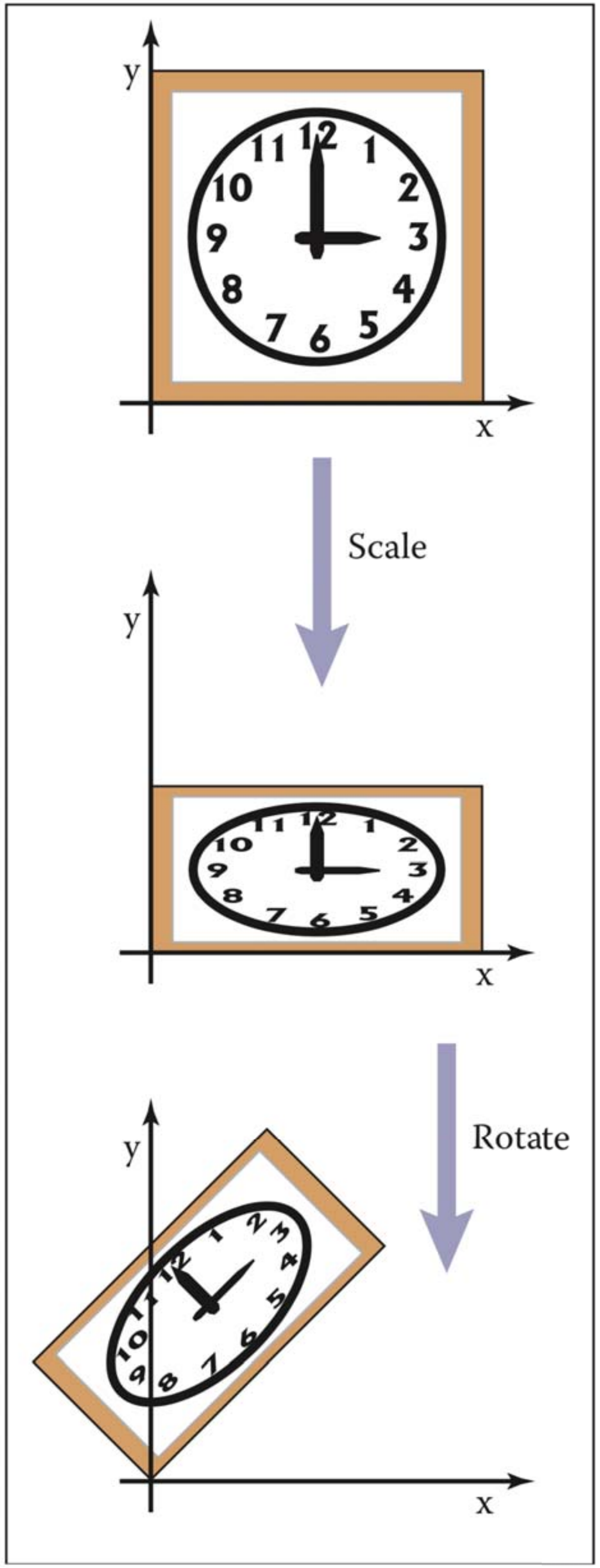
$$\begin{bmatrix} 0.707 & -0.707 \\ 0.707 & 0.707 \end{bmatrix}$$

- Scale:

$$\begin{bmatrix} 1 & 0 \\ 0 & 0.5 \end{bmatrix}$$

- Combined:

$$\begin{bmatrix} 0.707 & -0.707 \\ 0.707 & 0.707 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 0.5 \end{bmatrix} = \begin{bmatrix} 0.707 & -0.353 \\ 0.707 & 0.353 \end{bmatrix}$$



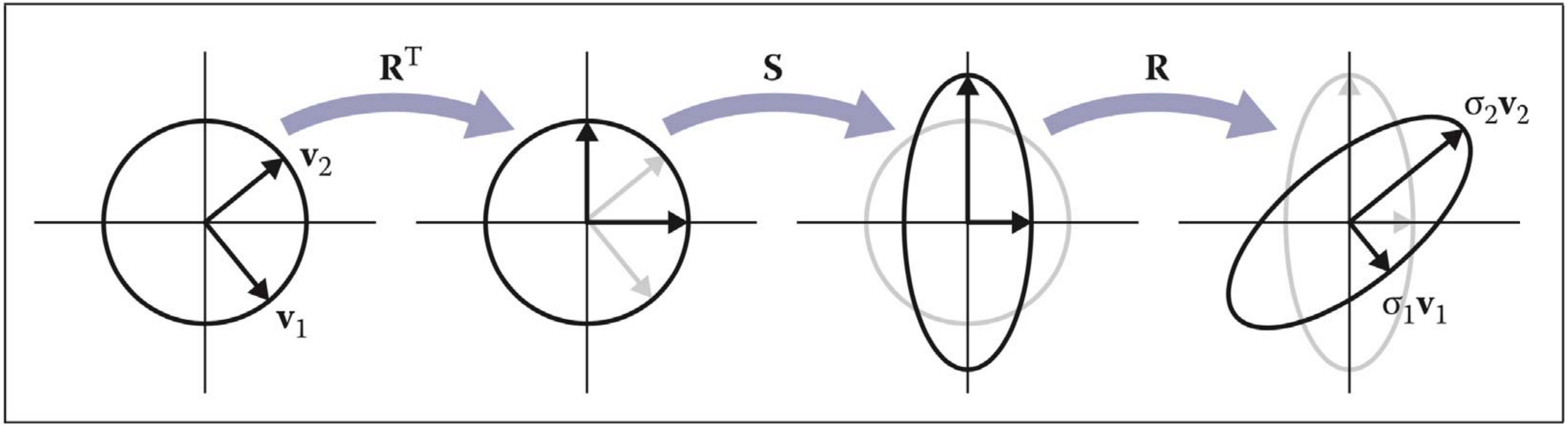
Decomposition

- Recall: a **symmetric matrix** is any matrix M such that $M = M^T$
- Let's consider a special type of composition, of the form RSR^T
  - Motivation: scaling on arbitrary axis, e.g. rotate, then scale, then rotate back
  - Example: $\mathrm{rotate}(-45^\circ)\mathrm{scale}(1.5, 1)\mathrm{rotate}(45^\circ)$

$$\begin{bmatrix} 1.25 & -0.25 \\ -0.25 & 1.25 \end{bmatrix}$$

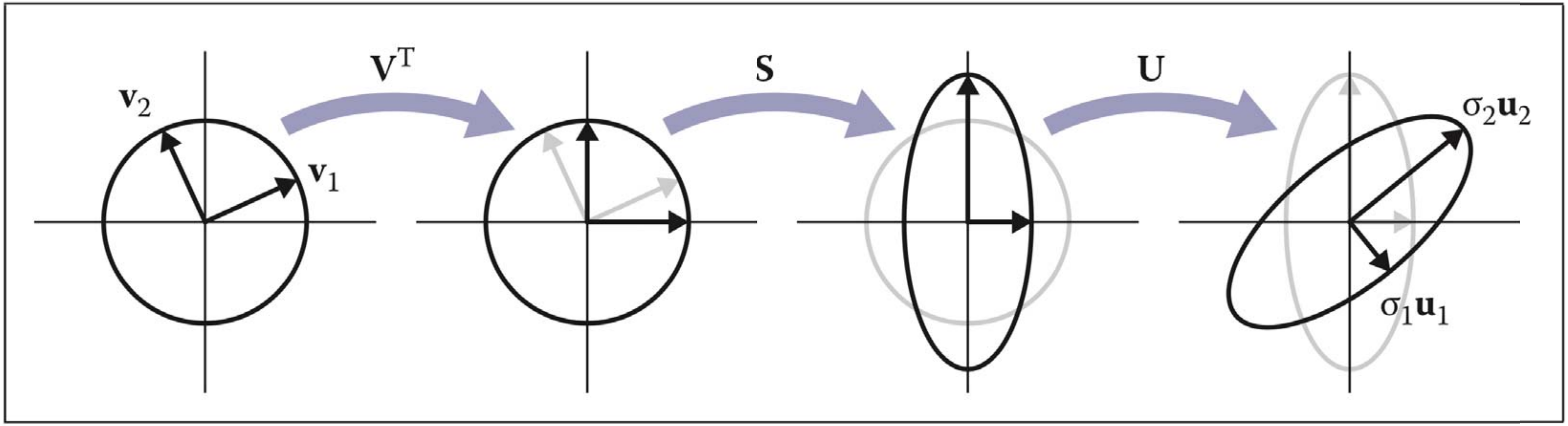
Decomposition

- RSR^T is always a symmetric matrix
  - Why? $(RSR^T)^T = R^{TT}S^TR^T = RSR^T$)
- Any symmetric metric A can be decomposed to $A = RSR^T$
  - All rotations R, happen to be **orthogonal matrices.**
- Any symmetric transformation matrix is a scale in some axis



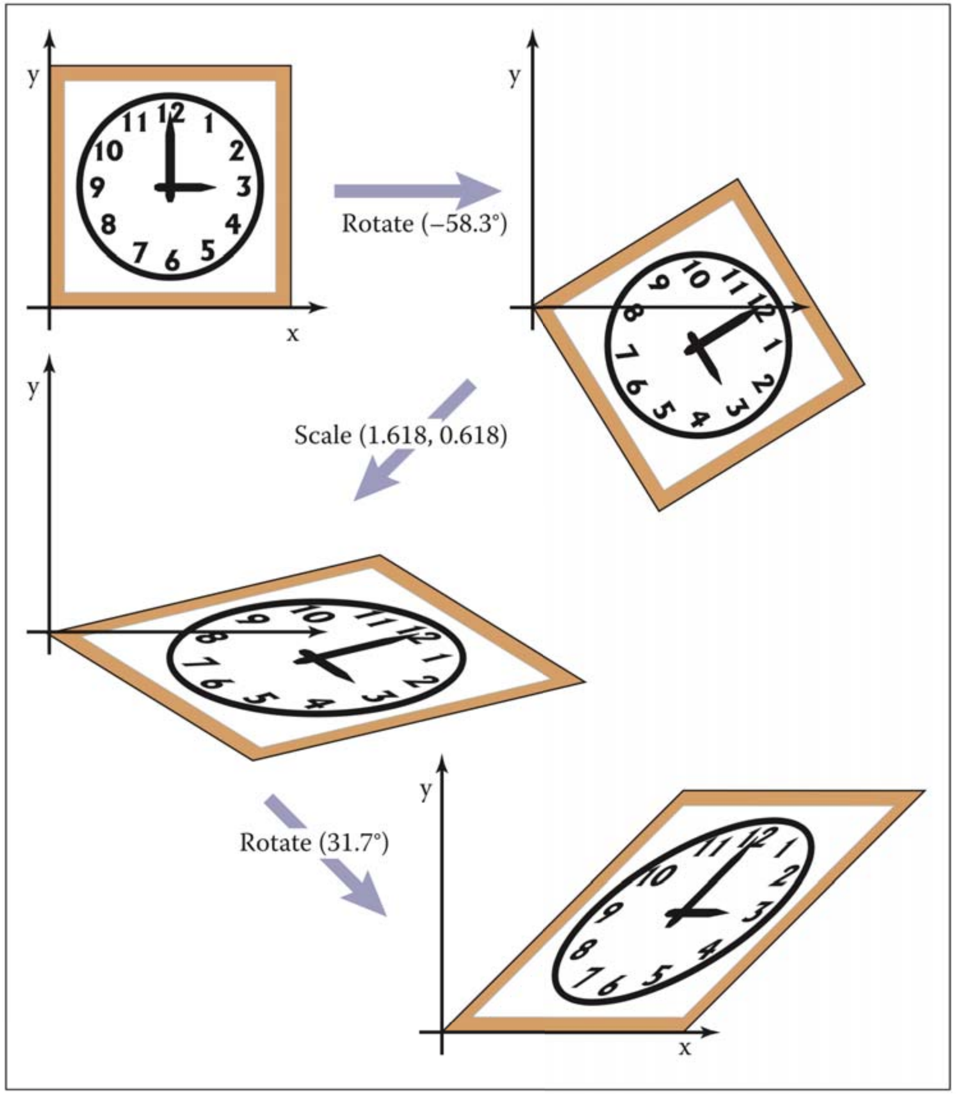
Decomposition

- Any arbitrary matrix can be decomposed to A = USV^T where U and V are orthogonal matrices (but not necessarily rotations)
  - This is called **singular value decomposition**
- This has a similar interpretation, but with different rotations before/after the scale



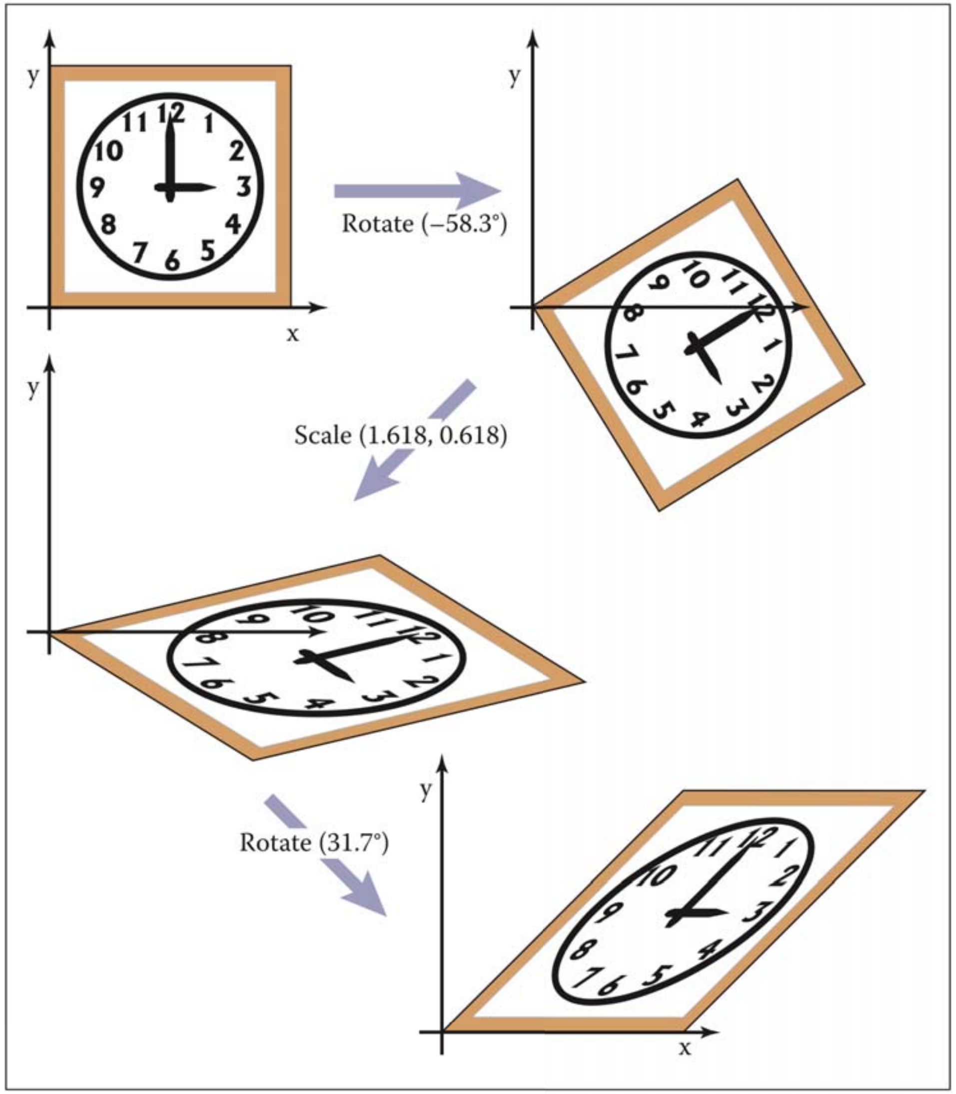
Some Consequences

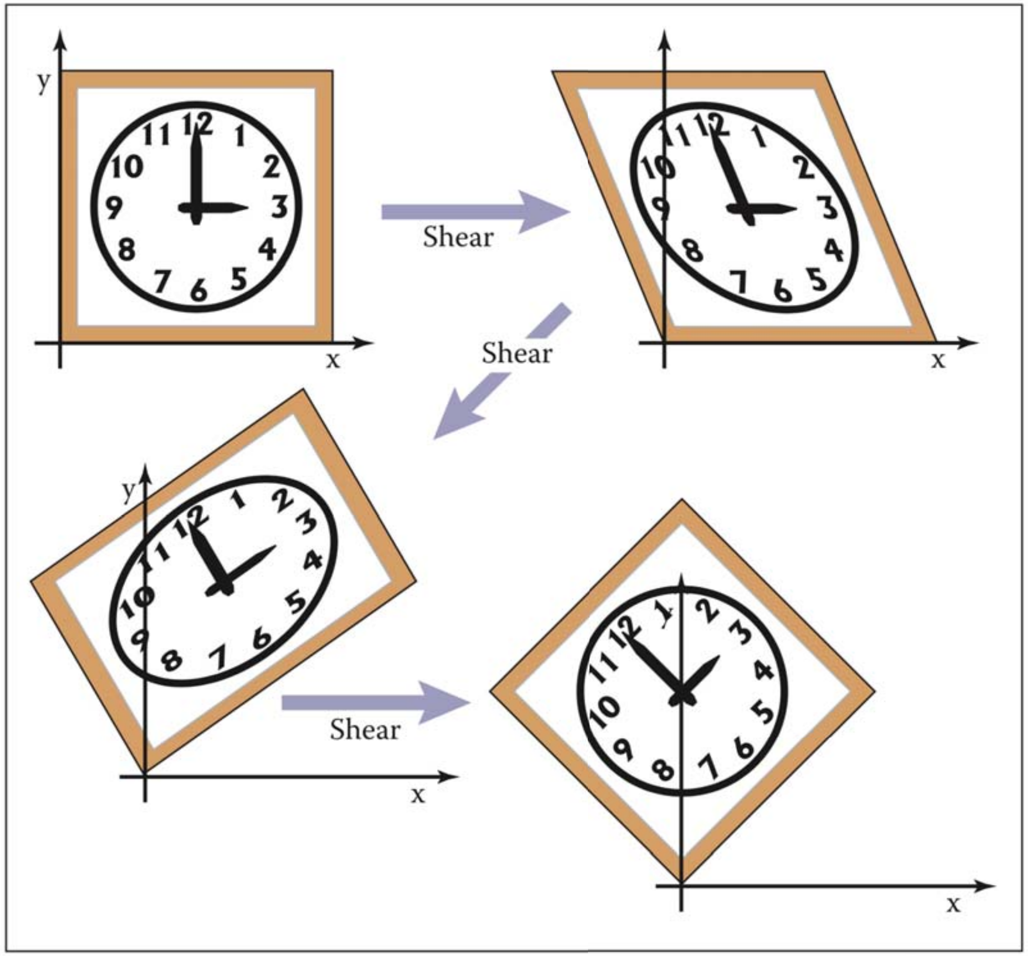
All shears are USV^T



Some Consequences

All shears are USV^T





Any rotation is three shears (Paeth decomposition)

Inversion

- Interpreting a matrix M as a geometric transformation, how might we undo the operation?

- We can apply the **inverse** transformation, it turns out by applying the inverse matrix, M^{-1}

  - Recall that $MM^{-1} = M^{-1}M = \mathrm{I}$, the identity matrix

- This is true for all operations we've discussed, e.g. scale by s is undone by scale of 1/s, rotate by angle ϕ is undone by rotate of angle $-\phi$

- <u>**Key point:**</u> the algebraic inverse *is* the geometric inverse

3D Linear Transformations

3D Linear Transformations

- We can transform points in a 3D coordinate system by multiplying the point (a vector) by a matrix (the transformation), just like in 2D!

- The only difference is we will use 3x3 matrices A by **x** = (x,y,z), or A**x**, e.g. for scale and shear:

$$\text{scale}(s_x, s_y, s_z) = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & s_z \end{bmatrix}$$

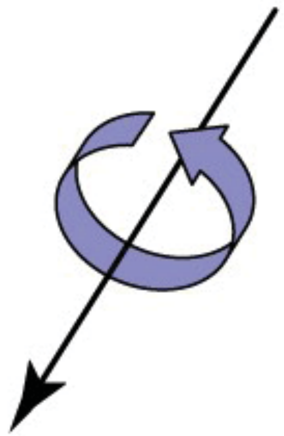
$$\text{shear-x}(d_y, d_z) = \begin{bmatrix} 1 & d_y & d_z \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Rotations in 3D

- In 2D, a rotation is about a point
- In 3D, a rotation is about an axis

convention: positive rotation is CCW

2D



3D

convention: positive rotation is CCW when axis vector is pointing at you

Rotations about 3D Axes

- In 3D, we need to pick an axis to rotate about

$$\text{rotate-z}(\phi) = \begin{bmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

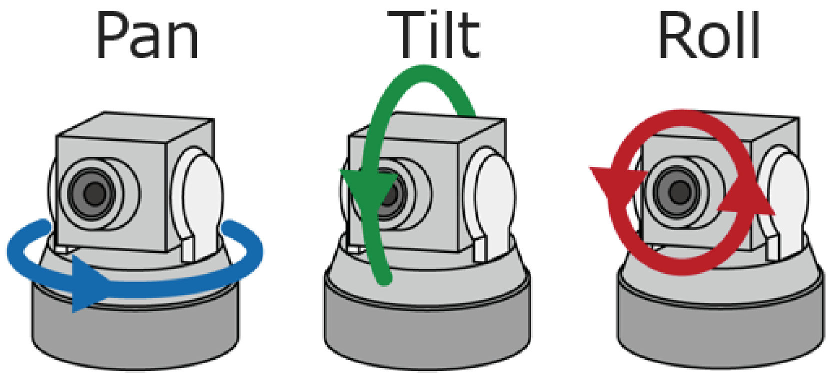
- And we can pick any of the three axes

$$\text{rotate-x}(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix}$$

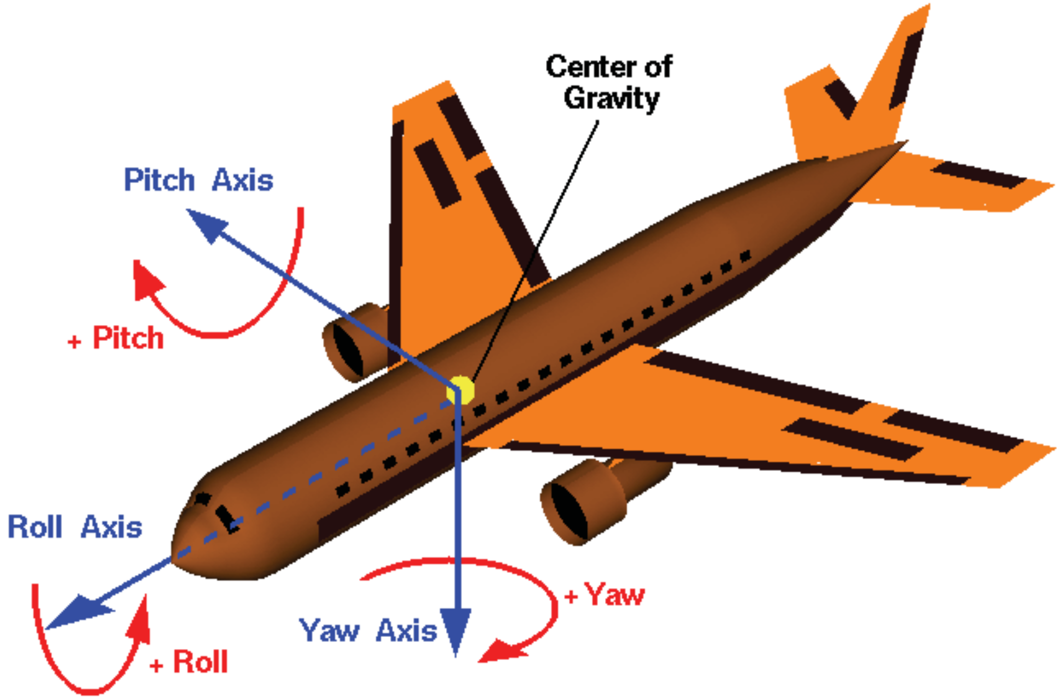
$$\text{rotate-y}(\phi) = \begin{bmatrix} \cos\phi & 0 & \sin\phi \\ 0 & 1 & 0 \\ -\sin\phi & 0 & \cos\phi \end{bmatrix}$$

Building Complex Rotations from Axis-Aligned Rotations

- Rotations about **x, y, z** are sometimes called **Euler angles**
- Build a combined rotation using matrix composition



Ishikawa Watanabe Laboratory



Wikipedia

Arbitrary Rotations

- To rotate about any axis: we change the coordinate space we are working in, using orthogonal matrices.
- Consider orthogonal matrix R_{uvw}, form by taking three orthogonal vectors $\mathbf{u}$, $\mathbf{v}$, and $\mathbf{w}$:

Arbitrary Rotations

- To rotate about any axis: we change the coordinate space we are working in, using orthogonal matrices.
- Consider orthogonal matrix R_{uvw}, form by taking three orthogonal vectors $\mathbf{u}$, $\mathbf{v}$, and $\mathbf{w}$:

Property of orthogonal vectors:

$$\mathbf{u} \cdot \mathbf{u} = \mathbf{v} \cdot \mathbf{v} = \mathbf{w} \cdot \mathbf{w} = 1$$

$$\mathbf{u} \cdot \mathbf{v} = \mathbf{v} \cdot \mathbf{w} = \mathbf{w} \cdot \mathbf{u} = 0$$

Arbitrary Rotations

- To rotate about any axis: we change the coordinate space we are working in, using orthogonal matrices.
- Consider orthogonal matrix R_{uvw}, form by taking three orthogonal vectors $\mathbf{u}$, $\mathbf{v}$, and $\mathbf{w}$:

Property of orthogonal vectors:

$$\mathbf{u} \cdot \mathbf{u} = \mathbf{v} \cdot \mathbf{v} = \mathbf{w} \cdot \mathbf{w} = 1$$
$$\mathbf{u} \cdot \mathbf{v} = \mathbf{v} \cdot \mathbf{w} = \mathbf{w} \cdot \mathbf{u} = 0$$

$$\mathrm{R}_{uvw} = \begin{bmatrix} \mathbf{u} \\ \mathbf{v} \\ \mathbf{w} \end{bmatrix}$$

Arbitrary Rotations

- To rotate about any axis: we change the coordinate space we are working in, using orthogonal matrices.

- Consider orthogonal matrix R_{uvw}, form by taking three orthogonal vectors $\mathbf{u}$, $\mathbf{v}$, and $\mathbf{w}$:

Property of orthogonal vectors:

$$\mathbf{u} \cdot \mathbf{u} = \mathbf{v} \cdot \mathbf{v} = \mathbf{w} \cdot \mathbf{w} = 1$$
$$\mathbf{u} \cdot \mathbf{v} = \mathbf{v} \cdot \mathbf{w} = \mathbf{w} \cdot \mathbf{u} = 0$$

$$\mathrm{R}_{uvw} = \begin{bmatrix} x_u & y_u & z_u \\ x_v & y_v & z_v \\ x_w & y_w & z_w \end{bmatrix}$$

Arbitrary Rotations

- What happens when we apply R_{uvw} to any of the basis vectors, e.g.:

$$\mathbf{R}_{uvw}\mathbf{u} = \begin{bmatrix} \mathbf{u}\cdot\mathbf{u} \\ \mathbf{v}\cdot\mathbf{u} \\ \mathbf{w}\cdot\mathbf{u} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \mathbf{x}$$

- But this means that if we apply $R_{uvw}^{\mathbf{T}}$ to the Cartesian coordinate vectors, e.g.:

$$\mathbf{R}_{uvw}^{\mathrm{T}}\mathbf{y} = \begin{bmatrix} x_u & x_v & x_w \\ y_u & y_v & y_w \\ z_u & z_v & z_w \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} x_v \\ y_v \\ z_v \end{bmatrix} = \mathbf{v}$$

Arbitrary Rotations

- This means that if we want to rotation around an arbitrary axis, we need only to use a change of coordinates

- E.g. to rotate around a direction $\mathbf{w}$, we
  - Compute orthogonal directions $\mathbf{u}$, $\mathbf{v}$, and $\mathbf{w}$
  - Change the $\mathbf{uvw}$ axes to be **xyz** (R_{uvw})
  - Apply a rotate-z()
  - Finally, change the axes back to $\mathbf{uvw}$ ($R_{uvw}^{\mathbf{T}}$)

$$\underbrace{\begin{bmatrix} x_u & x_v & x_w \\ y_u & y_v & y_w \\ z_u & z_v & z_w \end{bmatrix}}_{R_{uvw}^{\mathbf{T}}} \underbrace{\begin{bmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{\text{rotate-z()}} \underbrace{\begin{bmatrix} x_u & y_u & z_u \\ x_v & y_v & z_v \\ x_w & y_w & z_w \end{bmatrix}}_{R_{uvw}}$$

Transformations for Shapes in Computer Graphics

- These transformations also apply to transforming geometric objects

Transformations for Shapes in Computer Graphics

- These transformations also apply to transforming geometric objects
- Parametric forms:

Transformations for Shapes in Computer Graphics

- These transformations also apply to transforming geometric objects
- Parametric forms:
  - We can transform the points directly, e.g. M(**p**(t)) to transform the parametric positions **p**(t)

Transformations for Shapes in Computer Graphics

- These transformations also apply to transforming geometric objects
- Parametric forms:
  - We can transform the points directly, e.g. M(**p**(t)) to transform the parametric positions **p**(t)
- Implicit forms:

Transformations for Shapes in Computer Graphics

- These transformations also apply to transforming geometric objects
- Parametric forms:
    - We can transform the points directly, e.g. $M(\mathbf{p}(t))$ to transform the parametric positions $\mathbf{p}(t)$
- Implicit forms:
    - We invert the transform and test the predicate, e.g. if $f(M^{-1}(\mathbf{p})) = 0$ then $\mathbf{p}$ is on the transformed implicit shape