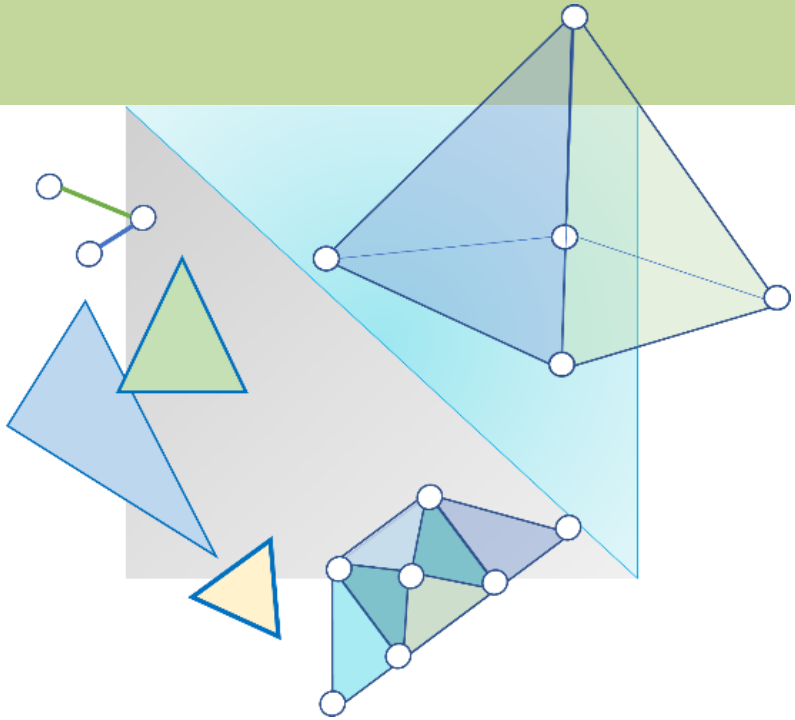
CITS3003 Graphics & Animation

Lecture 9: Transformations and Homogeneous Coordinates



Content

- Look in more detail into standard transformations
  - Rotation
  - Translation
  - Scaling
  - Shear
- Learn to build arbitrary transformation matrices from simple transformations

Matrix Multiplication

$$\begin{bmatrix} 1 & 1.1 \\ 2 & 2.2 \\ 3 & 3.3 \end{bmatrix} \times \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} 1a + 1.1b \\ 2a + 2.2b \\ 3a + 3.3b \end{bmatrix}$$

$$= a\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} + b\begin{bmatrix} 1.1 \\ 2.2 \\ 3.3 \end{bmatrix}$$

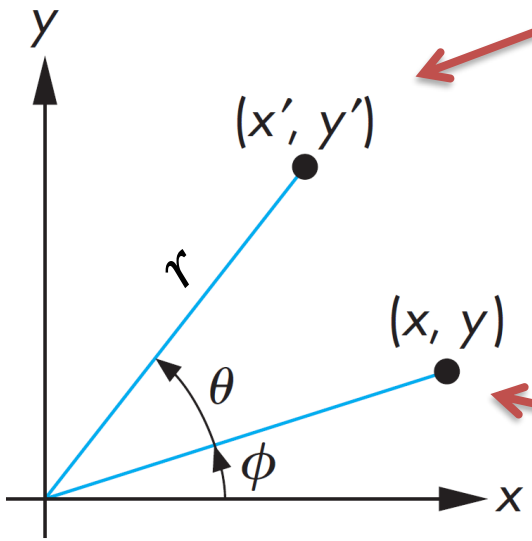
$a = b = 1$

$$= \begin{bmatrix} 2.1 \\ 4.2 \\ 6.3 \end{bmatrix}$$

$$= \begin{bmatrix} 2.1 \\ 4.2 \\ 6.3 \end{bmatrix}$$

Rotation (2D)

- Consider a rotation about the origin by θ degrees
  - radius stays the same, angle increases by θ



Two-dimensional rotation

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

(New point $\mathbf{p}'$ after rotation)

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

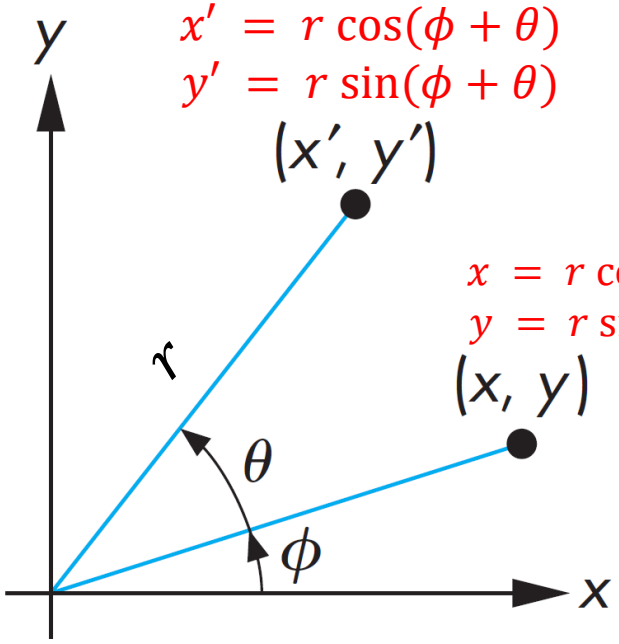
Representing x,y in the Polar form

(Old point $\mathbf{p}$ before rotation)

sin(A+B) = sin(A) cos(B) + cos(A) sin(B)
cos(A+B) = cos(A) cos(B) – sin(A) sin(B)

Rotation (2D)

- Consider a rotation about the origin by θ degrees
  - radius stays the same, angle increases by θ



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

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

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

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

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

Two-dimensional rotation

sin(A+B) = sin(A) cos(B) + cos(A) sin(B)
cos(A+B) = cos(A) cos(B) – sin(A) sin(B)

Rotation about the z-axis

- Rotation in two dimensions is equivalent to rotation about the z axis in three dimensions.
- Rotation about the z axis in 3D leaves the z components of all the points unchanged:

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

Rotation about the z-axis

$$R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \text{ x } \begin{bmatrix} x \\ y \\ 0 \\ 1 \end{bmatrix}$$

$$\begin{aligned} x' &= x\cos\theta - y\sin\theta + 0 + 0 &&= x\cos\theta - y\sin\theta \\ y' &= x\sin\theta + y\cos\theta + 0 + 0 &&= x\sin\theta + y\cos\theta \\ z' &= 0 + 0 + 0 + 0 &&= 0 \\ w' &= 0 + 0 + 0 + 1 &&= 1 \end{aligned}$$

In homogeneous coordinates

$$\mathbf{p}' = \mathbf{R}_Z(\theta)\mathbf{p}$$

Rotation about the z-axis (cont.)

- Consider the example

$$x = r\cos\phi$$
$$y = r\sin\phi$$
$$z = 0$$

Representing x,y in the Polar form

we have

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = R_z(\theta) \begin{bmatrix} r\cos\phi \\ r\sin\phi \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r\cos\phi \\ r\sin\phi \\ 0 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} r\cos\theta\cos\phi - r\sin\theta\sin\phi \\ r\sin\theta\cos\phi + r\cos\theta\sin\phi \\ 0 \\ 1 \end{bmatrix}$$

Rotation about the z-axis (cont.)

- Applying the rule:

$$\cos(\theta + \phi) = \cos\theta\cos\phi - \sin\theta\sin\phi$$
$$\sin(\theta + \phi) = \sin\theta\cos\phi + \cos\theta\sin\phi$$

we get

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = R_z(\theta) \begin{bmatrix} r\cos\phi \\ r\sin\phi \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} r\cos(\theta + \phi) \\ r\sin(\theta + \phi) \\ 0 \\ 1 \end{bmatrix}$$

- Thus,
  - $x' = r\cos(\theta + \phi)$
  - $y' = r\sin(\theta + \phi)$
  - $z' = 0$

More commonly,

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

Rotation about x and y axes

- Same argument as for rotation about *z* axis
  - For rotation about *x* axis, *x* is unchanged
  - For rotation about *y* axis, *y* is unchanged

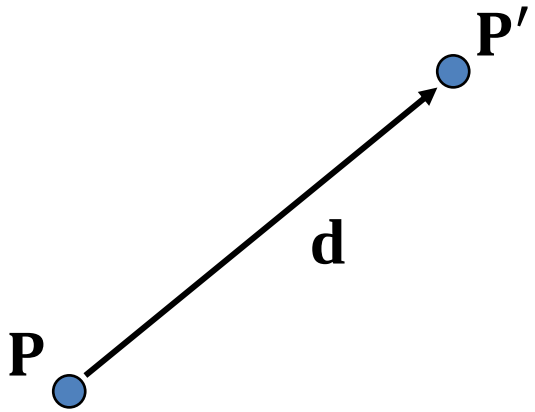
$$\mathbf{R} = \mathbf{R}_x(q) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{R} = \mathbf{R}_y(q) = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Note the negative sign here

Translation

- Move (translate, displace) a point to a new location

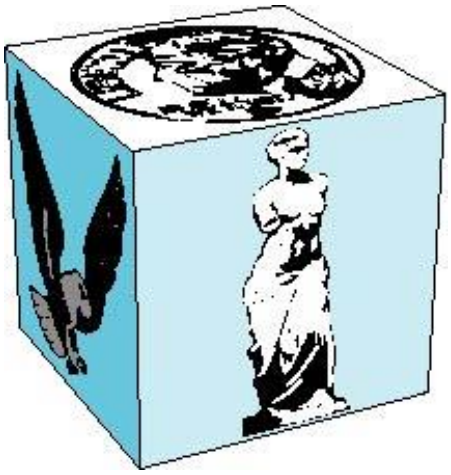


- Displacement determined by a vector **d**
  - Three degrees of freedom

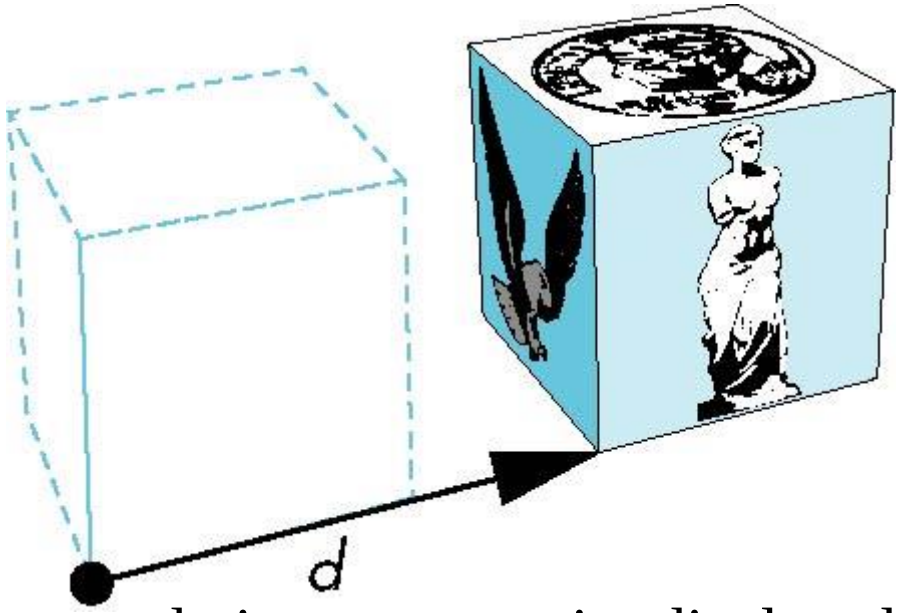
$$\mathbf{P}' = \mathbf{P} + \mathbf{d}$$

The whole object moves

- Although we can move a point to a new location in an infinite number of ways, when we move many points of a rigid object there is usually only one way



object



translation: every point displaced by same vector

Translation using Representations

- Using the homogeneous coordinate representation in some frame
  - $\mathbf{p} = [x \quad y \quad z \quad 1]^T$
  - $\mathbf{p}' = [x' \quad y' \quad z' \quad 1]^T$
  - $\mathbf{d} = [d_x \quad d_y \quad d_z \quad 0]^T$
- Hence $\mathbf{p}' = \mathbf{p} + \mathbf{d}$ or
  - $x' = x + d_x$
  - $y' = y + d_y$
  - $z' = z + d_z$

note that this expression is in four dimensions and expresses point = vector + point

Translation Matrix

- We can also express translation using a 4×4 matrix $\mathbf{T}$ in homogeneous coordinates

$\mathbf{p}' = \mathbf{T}\mathbf{p}$ where

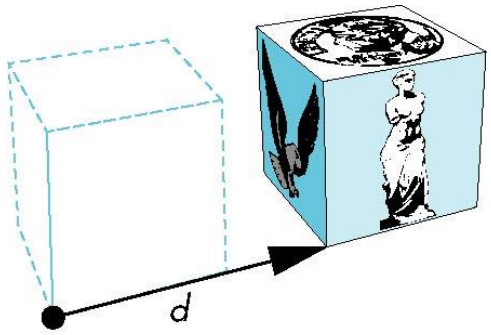
$$\mathbf{T} = \mathbf{T}(d_x, d_y, d_z) = \begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- This form is better for implementation because all affine transformations can be expressed this way and multiple transformations can be concatenated together

Translation

- Translate object= Move each vertex by the same distance d =(d_x, d_y, d_z)

$$\mathbf{T} = \mathbf{T}(d_x, d_y, d_z) = \begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



If we translate a point (2,2,2) by displacement (**2,4,6**), new location of point is (4,6,8)

Where

- $x' = x + d_x$
- $y' = y + d_y$
- $z' = z + d_z$

Using matrix multiplication for translation

$$\begin{pmatrix} 4 \\ 6 \\ 8 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 4 \\ 0 & 0 & 1 & 6 \\ 0 & 0 & 0 & 1 \end{pmatrix} \times \begin{pmatrix} 2 \\ 2 \\ 2 \\ 1 \end{pmatrix}$$

Translated point | **Translation Matrix** | **Original point**

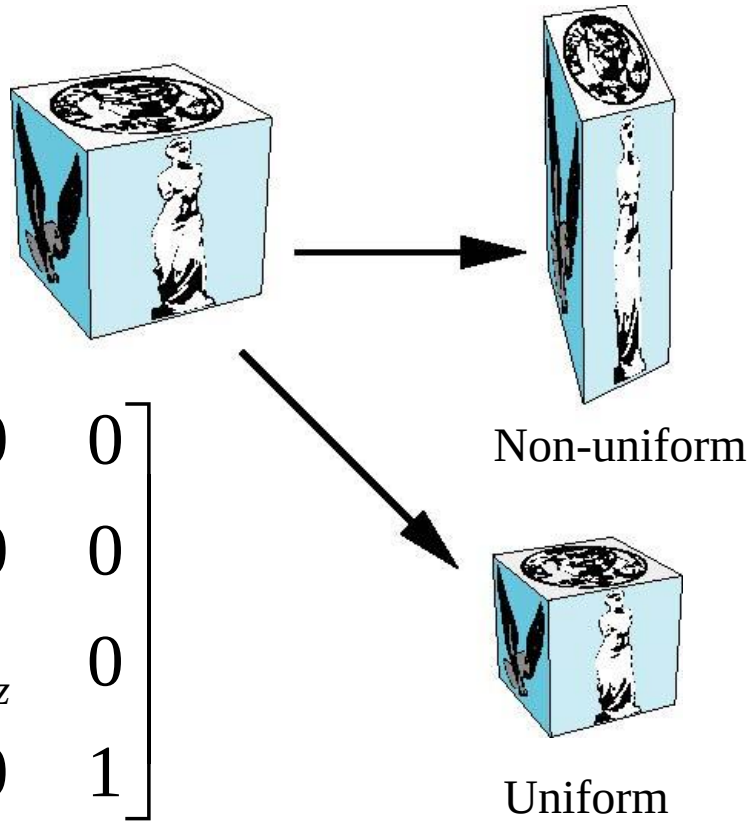
Scaling (non-rigid transform)

- Expand or contract along each axis (fixed point of origin)

$$x' = s_x x$$
$$y' = s_y y$$
$$z' = s_z z$$

$$\mathbf{p}' = \mathbf{S}\mathbf{p}$$

$$\mathbf{S} = \mathbf{S}(s_x, s_y, s_z) = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

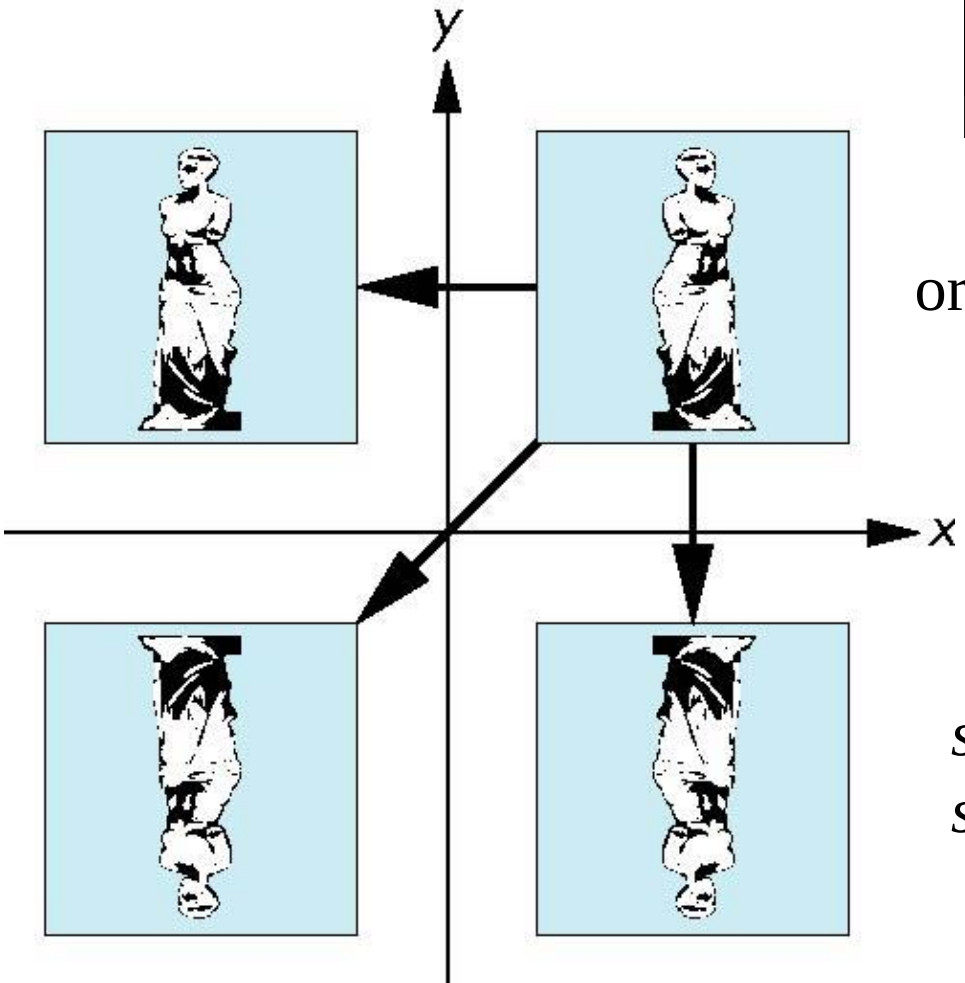


Reflection

corresponds to negative scale factors

$$\begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$s_x = -1$
$s_y = 1$



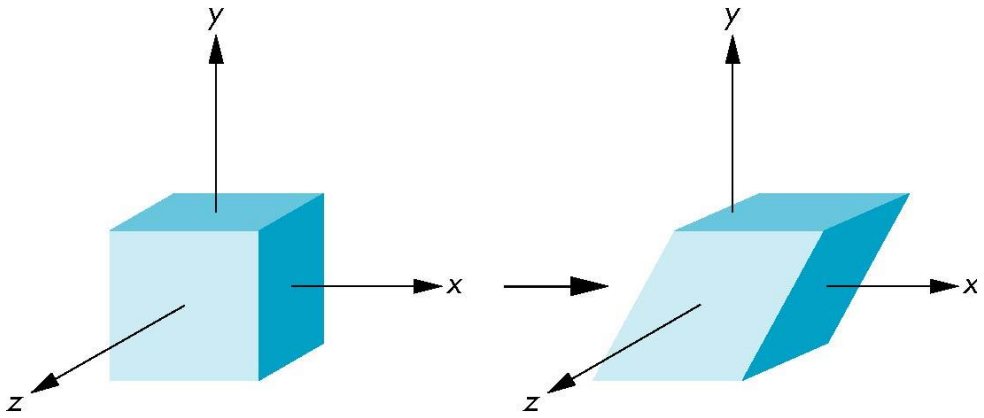
original

$s_x = -1$
$s_y = -1$

$s_x = 1$
$s_y = -1$

Shear

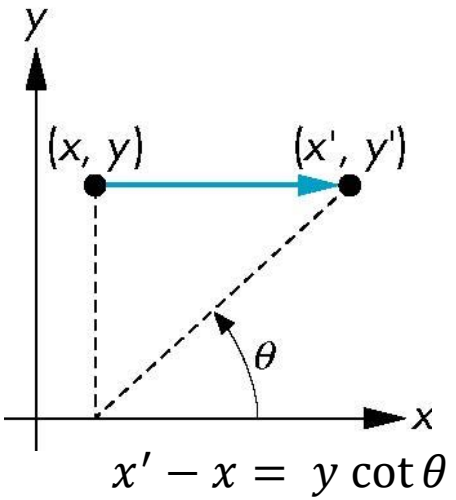
- It is helpful to add one more basic transformation, the ***shearing transformation***, to the collection of transformation we have learnt
- Shearing is equivalent to pulling faces in opposite directions



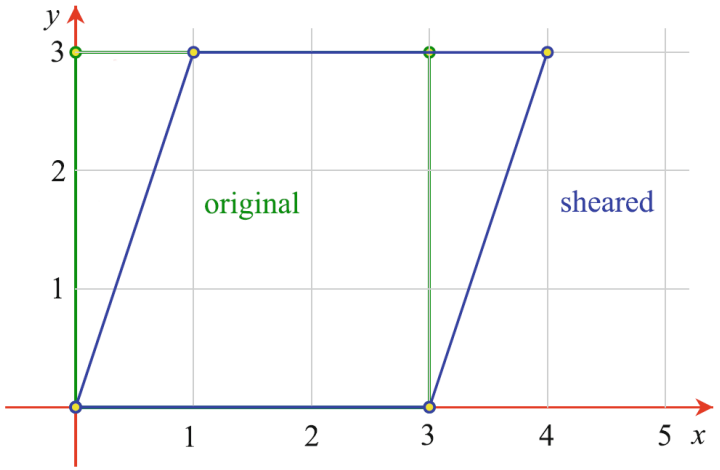
Shear

Consider a simple shear along the ***x* axis**

- $x' = x + y \cot\theta$
- $y' = y$
- $z' = z$



$$\Rightarrow \mathbf{H}(\theta) = \begin{bmatrix} 1 & \cot\theta & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

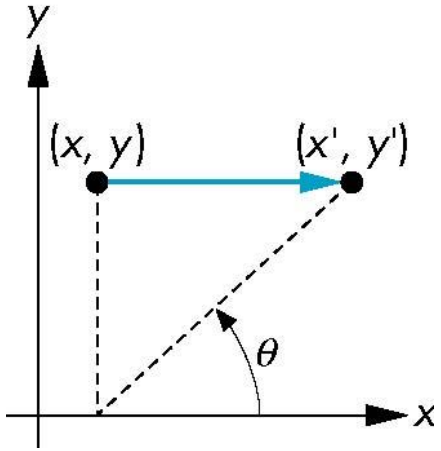


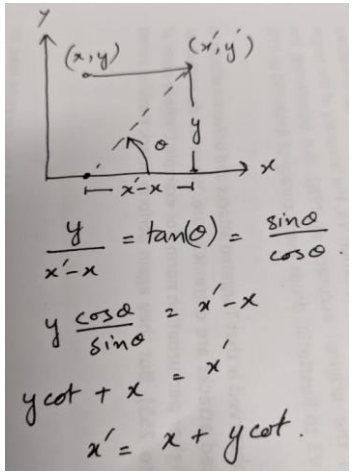
Shear

Consider a simple shear along the *x* axis

- $x' = x + y \cot\theta$
- $y' = y$
- $z' = z$

$$\Rightarrow \mathbf{H}(\theta) = \begin{bmatrix} 1 & \cot\theta & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$





Inverses

- Although we could compute inverse matrices by general formulas, we can use simple geometric observations
  - **Translation**: $\mathbf{T}^{-1}(d_x, d_y, d_z) = \mathbf{T}(-d_x, -d_y, -d_z)$
  - **Rotation**: $\mathbf{R}^{-1}(\theta) = \boldsymbol{R}(-\theta)$
    - Holds for any rotation matrix
    - Note that since $\cos(-\theta) = \cos(\theta)$ and $\sin(-\theta) = -\sin(\theta)$

$$\mathbf{R}^{-1}(\theta) = \mathbf{R}^{\mathrm{T}}(\theta)$$

  - **Scaling**: $\mathbf{S}^{-1}(s_x, s_y, s_z) = \mathbf{S}(1/s_x, 1/s_y, 1/s_z)$
  - **Shear**: $\mathbf{H}_x^{-1}(\theta) = \mathbf{H}_x(-\theta)$

Concatenation

- We can form arbitrary affine transformation matrices by multiplying together rotation, translation, scaling and shear matrices
- Because the same transformation is applied to many vertices, the cost of forming a matrix $\mathbf{M = ABCD}$ is not significant compared to the cost of computing $\mathbf{Mp}$ for many vertices $\mathbf{p}$
- The difficult part is how to form a desired transformation from the specifications in the application

Order of Transformations

- Note that matrix on the right is the first applied
- Mathematically, the following are equivalent

$$\mathbf{p}' = \mathbf{ABCp} = \mathbf{A}(\mathbf{B}(\mathbf{Cp}))$$

- Note many references use row vectors to represent points. For such references:

$$\mathbf{p}'^{\mathrm{T}} = \mathbf{p}^{\mathrm{T}}\mathbf{C}^{\mathrm{T}}\mathbf{B}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}$$

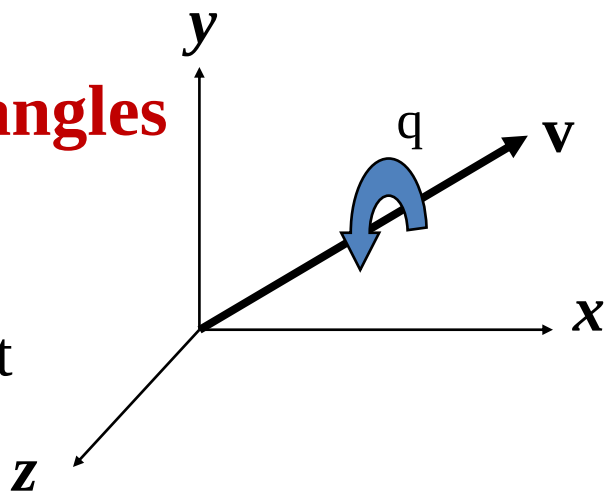
General Rotation About the Origin

A rotation angle of θ about an arbitrary axis can be decomposed into the concatenation of rotations about the x, y, and z axes

$$\boldsymbol{R}(\theta) = \boldsymbol{R}_z(\theta_z)\,\boldsymbol{R}_y(\theta_y)\,\boldsymbol{R}_x(\theta_x)$$

$\theta_x, \theta_y,$ and θ_z are called the **Euler angles**

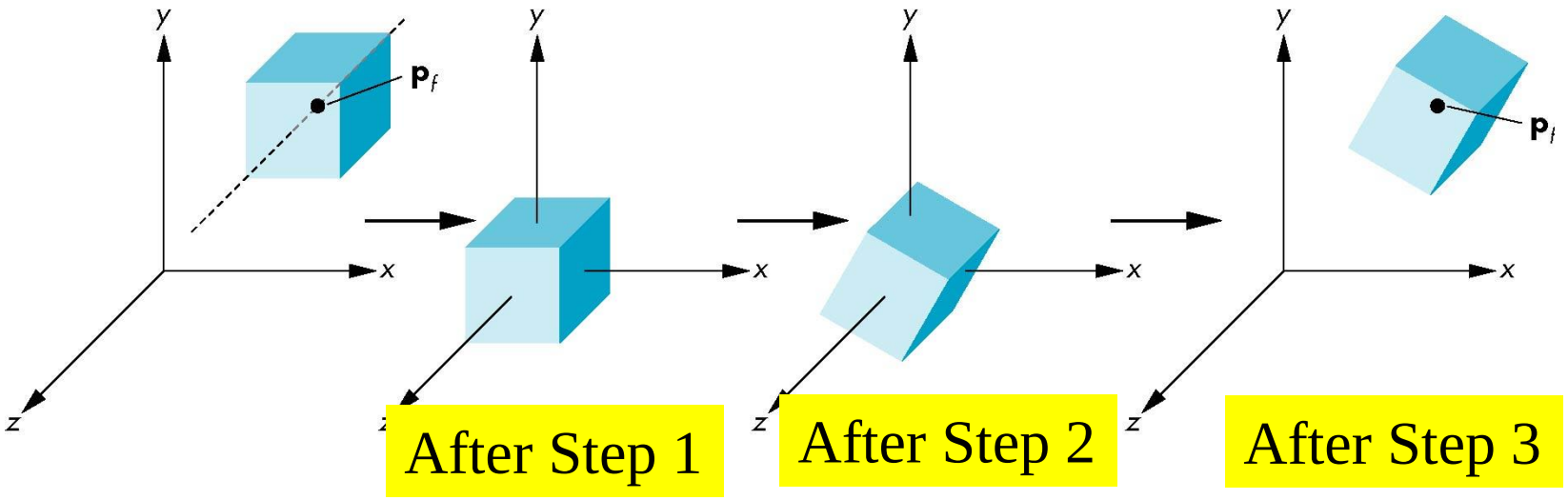
Note that **rotations do not commute**
We can use rotations in another order but with different angles



Rotation About a Fixed Point other than the Origin

1. Move the origin to the fixed point
2. Rotate
3. Move the origin back to fixed point back

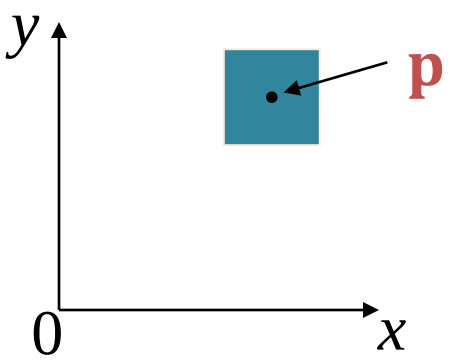
=> **M** = **T**(p_f) **R**(q) **T**(-p_f)



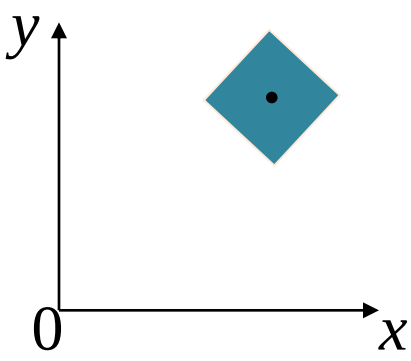
Rotation About a Fixed Point other than the Origin (cont.)

- A 2D example:

  Objective: want to rotate a square 45 degrees about its own center, **p**.



Before rotation

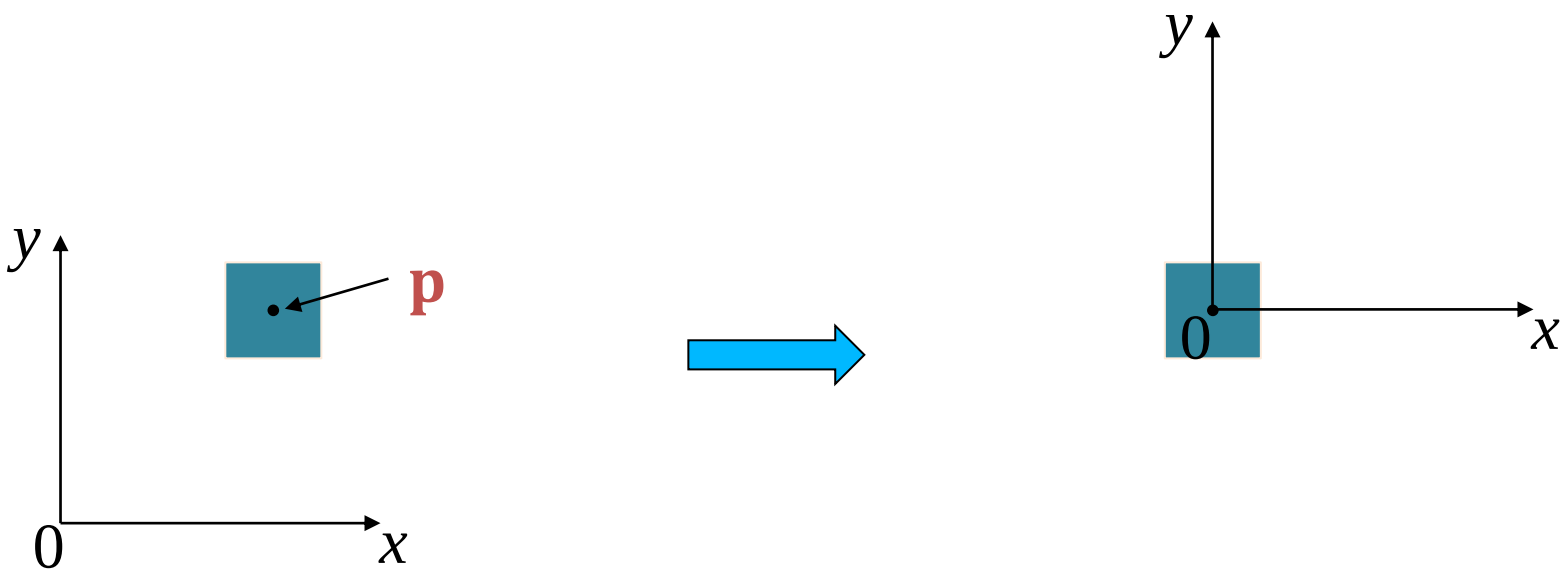


Output wanted after rotation

This is the same as rotating about the *z*-axis (pointing out of the page) in 3D.

Rotation About a Fixed Point other than the Origin (cont.)

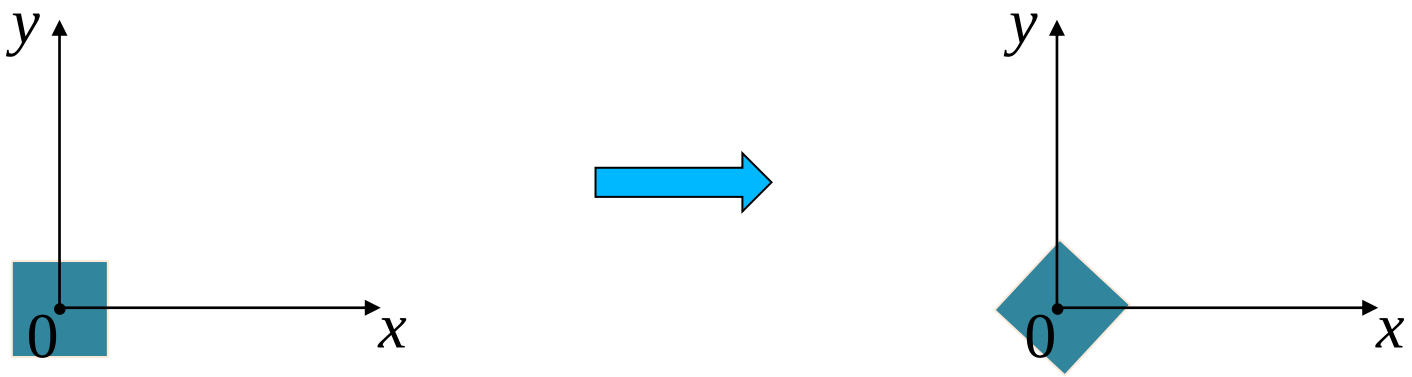
- Our aim is to construct a matrix **M** so that when the four vertices of the square are pre-multiplied by we get the desired output.
- **Step 1:** apply a translation so that the origin is at **p**.



$$\mathbf{M} = \mathbf{T}(-\mathbf{p})$$

Rotation About a Fixed Point other than the Origin (cont.)

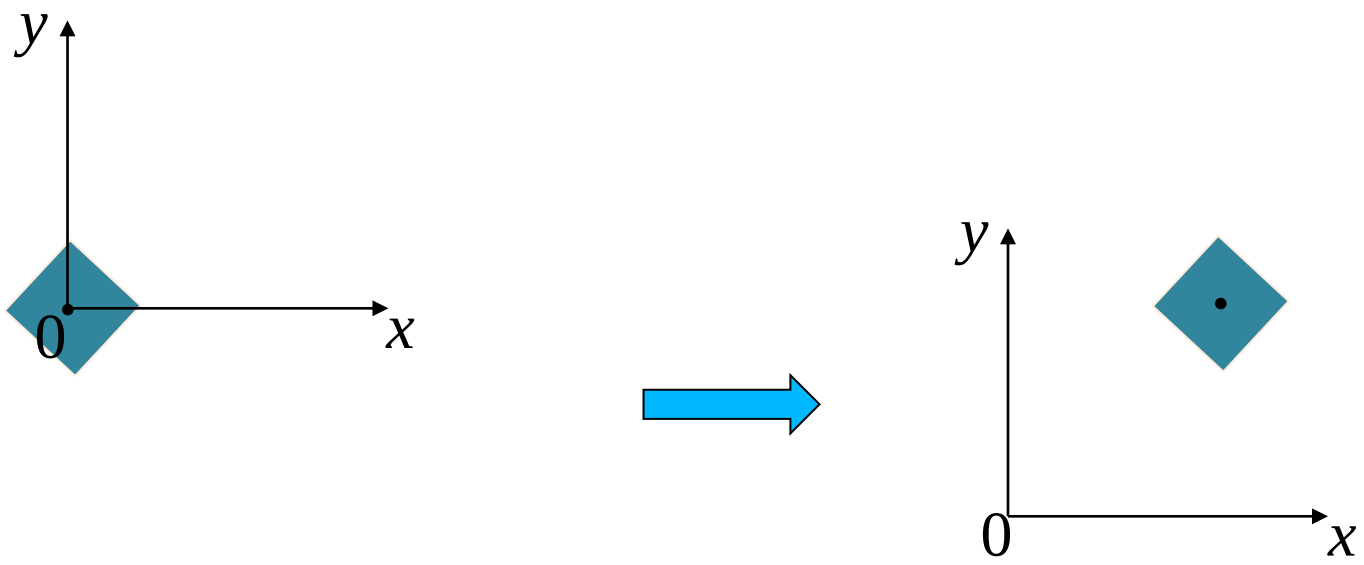
- **Step 2:** apply a 45 degree rotation about the z-axis at the origin.



$$\mathbf{M} = \mathbf{R}_z(\boldsymbol{\theta})\,\mathbf{T}(-\mathbf{p})$$

Rotation About a Fixed Point other than the Origin (cont.)

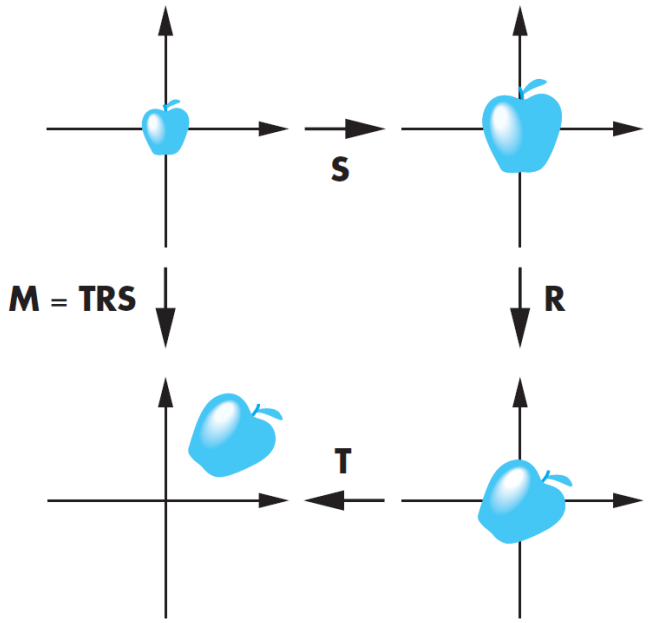
- **Step 3:** move the origin back to where it was before.



$$\mathbf{M} = \mathbf{T}(\mathbf{p})\,\mathbf{R}_z(\theta)\,\mathbf{T}(-\mathbf{p})$$

Instancing

- In modeling, we often start with a simple object centered at the origin, oriented with the axis, and at a standard size
- We apply an ***instance transformation*** to its vertices to
  - Scale
  - Orient
  - Locate



Further Reading

"Interactive Computer Graphics – A Top-Down Approach with Shader-Based OpenGL" by Edward Angel and Dave Shreiner, 6th Ed, 2012

- Sec 3.7 *Affine Transformations* (all subsections)
- Sec 3.8 *Translation, Rotation, and Scaling*
- Sec 3.9 *Transformations in Homogeneous Coordinates*
- Sec 3.10 *Concatenation of Transformations*

On Mid-term test

- **When**
  - Monday 04.04.2022 (01:00PM – 02:00PM)
- **Where**
  - Online (LMS + MSTeams)
    - Mid test will be released on LMS
    - Join MSTeams meeting (for attendance)
- **Mode**
  - Open-book, invigilated
- **Types of Questions**
  - Multiple Choice Questions (MCQ's)
  - True/False
- **Test duration**
  - 40 minutes
  - Expect to answer 1 – 1.5 questions per minute.

On Mid-term test

- If you have a clash with the test schedule, contact UC within this week

Example questions

Select all the correct statements

a) In a graphics pipeline, objects are processed one at a time, as passed by the application program

b) In parallel projection, all projected rays meet at the center of projection

c) Clipping must be performed before ‘primitive assembly’ in graphics pipeline because it can make ‘primitive assembly’ more efficient by reducing the number of primitives

d) Immediate mode graphics APIs are generally procedural

Example questions

Considering the common OpenGL program structure that consists of main(), init() and display(), which of the following statement(s) you would not expect to be present in the main() function when using GLUT

a) glutMainLoop();
b) glInitWindowSize(256,256);
c) init();
d) glutCreateWindow("anyName");
e) glDrawArrays(GL_TRIANGLES, 0, NumVertices);

Example questions

Minimizing the maximum interior angles of triangles makes triangles more suitable for rendering.

a) True

b) False

This is true because minimizing the max interior angle has the same effect as maximizing the minimum angle when the sum of the angles is always 180.