ΗΥ416
ΓΡΑΦΙΚΑ ΥΠΟΛΟΓΙΣΤΩΝ

Γεωμετρικοί Μετασχηματισμοί

Π. ΤΣΟΜΠΑΝΟΠΟΥΛΟΥ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ & ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Homogeneous Coordinates

- Add an extra dimension
  - in 2D, we use 3 x 3 matrices
  - In 3D, we use 4 x 4 matrices
- Each point has an extra value, *w*

$$\begin{pmatrix} x' \\ y' \\ z' \\ w' \end{pmatrix} = \begin{pmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix}$$

$$p' = M\,p$$

Homogeneous Coordinates

- Most of the time w = 1, and we can ignore it

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

Homogeneous Coordinates

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix}$$ can be represented as $$\begin{bmatrix} X \\ Y \\ Z \\ w \end{bmatrix}$$

where $$x = \frac{X}{w}, \quad y = \frac{Y}{w}, \quad z = \frac{Z}{w}$$

3-D Homogeneous Coordinates

- *Homogeneous coordinates*: represent coordinates in 3 dimensions with a 4-vector

$$(x, y, z) = \begin{bmatrix} x/w \\ y/w \\ z/w \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

  - $[x, y, z, 0]^T$ represents a point at infinity (use for vectors)
  - $[0, 0, 0]^T$ is not allowed
  - Note that typically w = 1 in object coordinates

More On Homogeneous Coords

- *What effect does the following matrix have?*

$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 10 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

- Conceptually, the fourth coordinate *w* is a bit like a scale factor

More On Homogeneous Coords

- Intuitively:
  - The *w* coordinate of a homogeneous point is typically 1
  - Decreasing *w* makes the point "bigger", meaning further from the origin
  - Homogeneous points with *w* = 0 are thus "points at infinity", meaning infinitely far away in some direction. *(What direction?)*
  - To help illustrate this, imagine subtracting two homogeneous points

3-D Homogeneous Coordinates

- Homogeneous coordinates seem unintuitive, but they make graphics operations *much* easier
- Our transformation matrices are now 4x4:
- Rotation around x-axis

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

3-D Homogeneous Coordinates

- Homogeneous coordinates seem unintuitive, but they make graphics operations *much* easier
- Our transformation matrices are now 4x4:
- Rotation around y-axis

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

3-D Homogeneous Coordinates

- Homogeneous coordinates seem unintuitive, but they make graphics operations *much* easier
- Our transformation matrices are now 4x4:
- Rotation around z-axis

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

3-D Homogeneous Coordinates

- Homogeneous coordinates seem unintuitive, but they make graphics operations *much* easier
- Our transformation matrices are now 4x4:

$$\mathbf{S} = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Performing a scale:

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

3-D Homogeneous Coordinates

- *How can we represent **translation** as a 4x4 matrix?*
- A: Using the rightmost column:

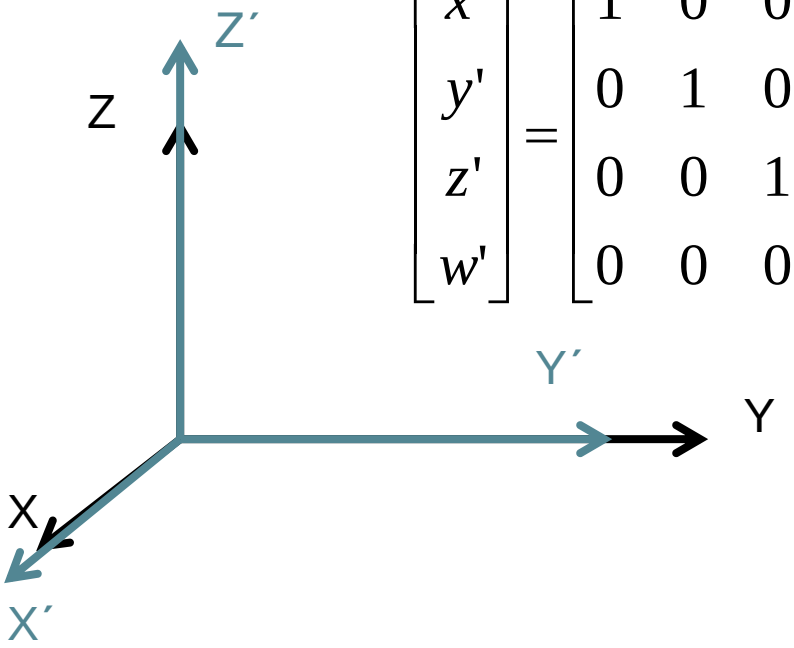
$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Performing a translation:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 10 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x & y & z+10 & 1 \end{bmatrix}$$

Translation Matrices

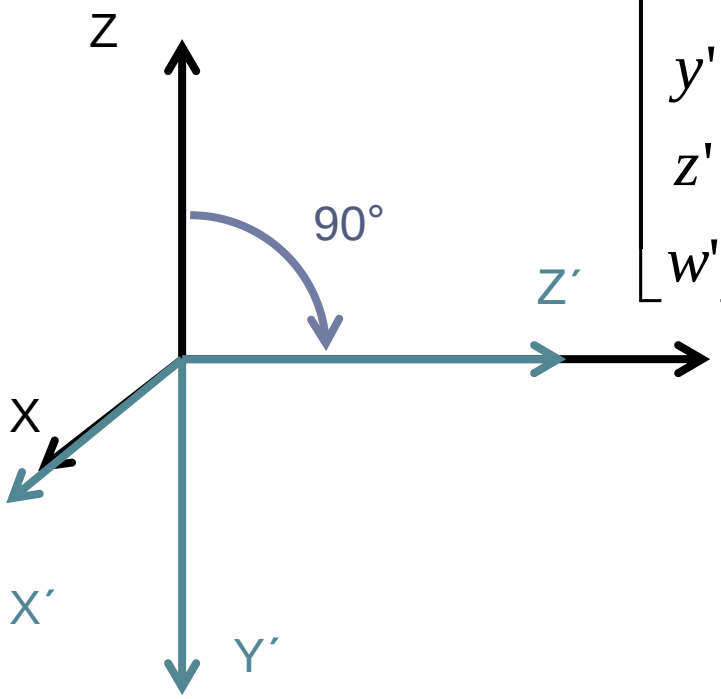
- Now that we can represent translation as a matrix, we can composite it with other transformations
- Ex: rotate **an point** 90° about **X**, then move it 10 units down (new)**Z-axis**:



$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -10 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(90°) & -\sin(90°) & 0 \\ 0 & \sin(90°) & \cos(90°) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

Translation Matrices

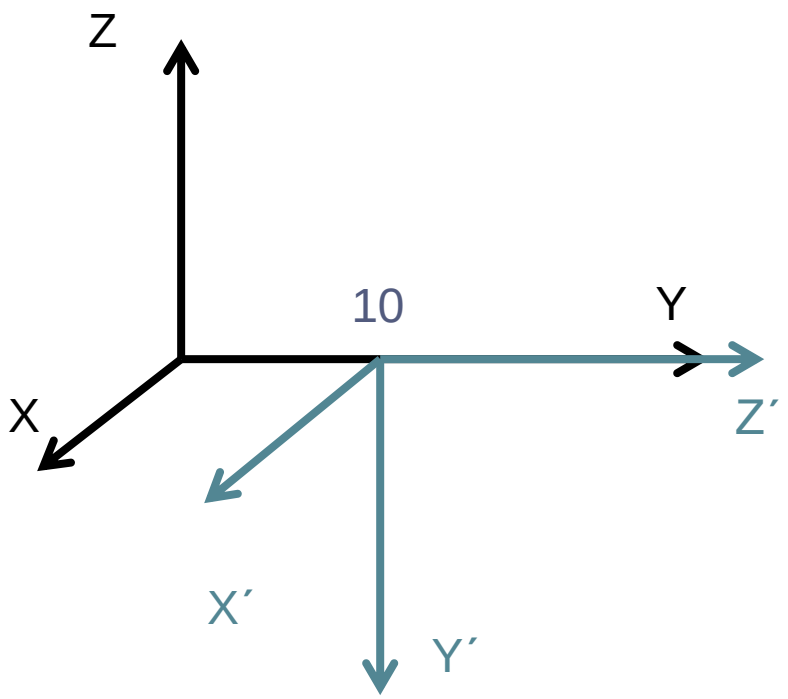
- Now that we can represent translation as a matrix, we can composite it with other transformations
- Ex: rotate **an point** 90° about **X**, then move it 10 units down (new)**Z-axis**:



$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -10 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

Translation Matrices

- Now that we can represent translation as a matrix, we can composite it with other transformations
- Ex: rotate **an point** 90° about **X**, then move it 10 units down (new)**Z-axis**:

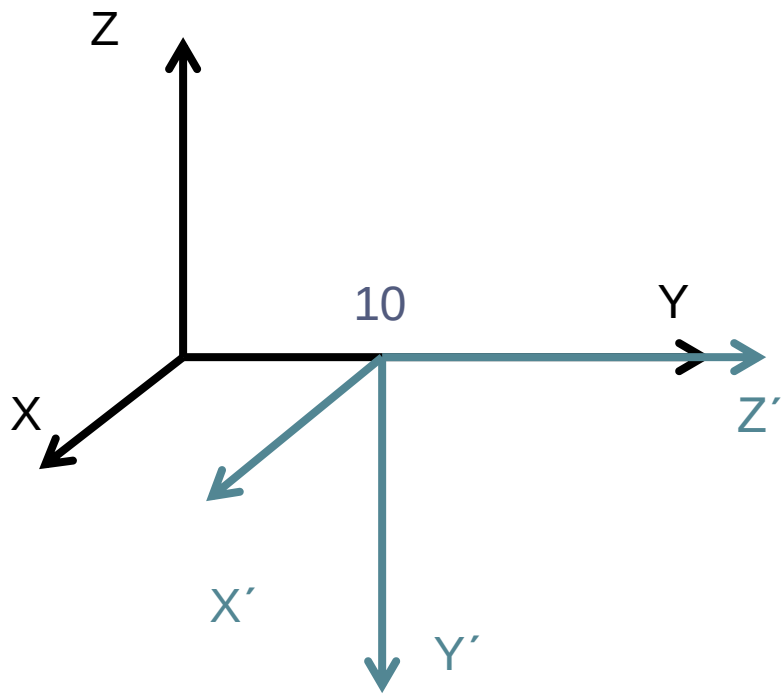


$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & -10 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

Translation Matrices

- Now that we can represent translation as a matrix, we can composite it with other transformations
- Ex: rotate **an point** 90° about **X**, then move it 10 units down (new)**Z-axis**:

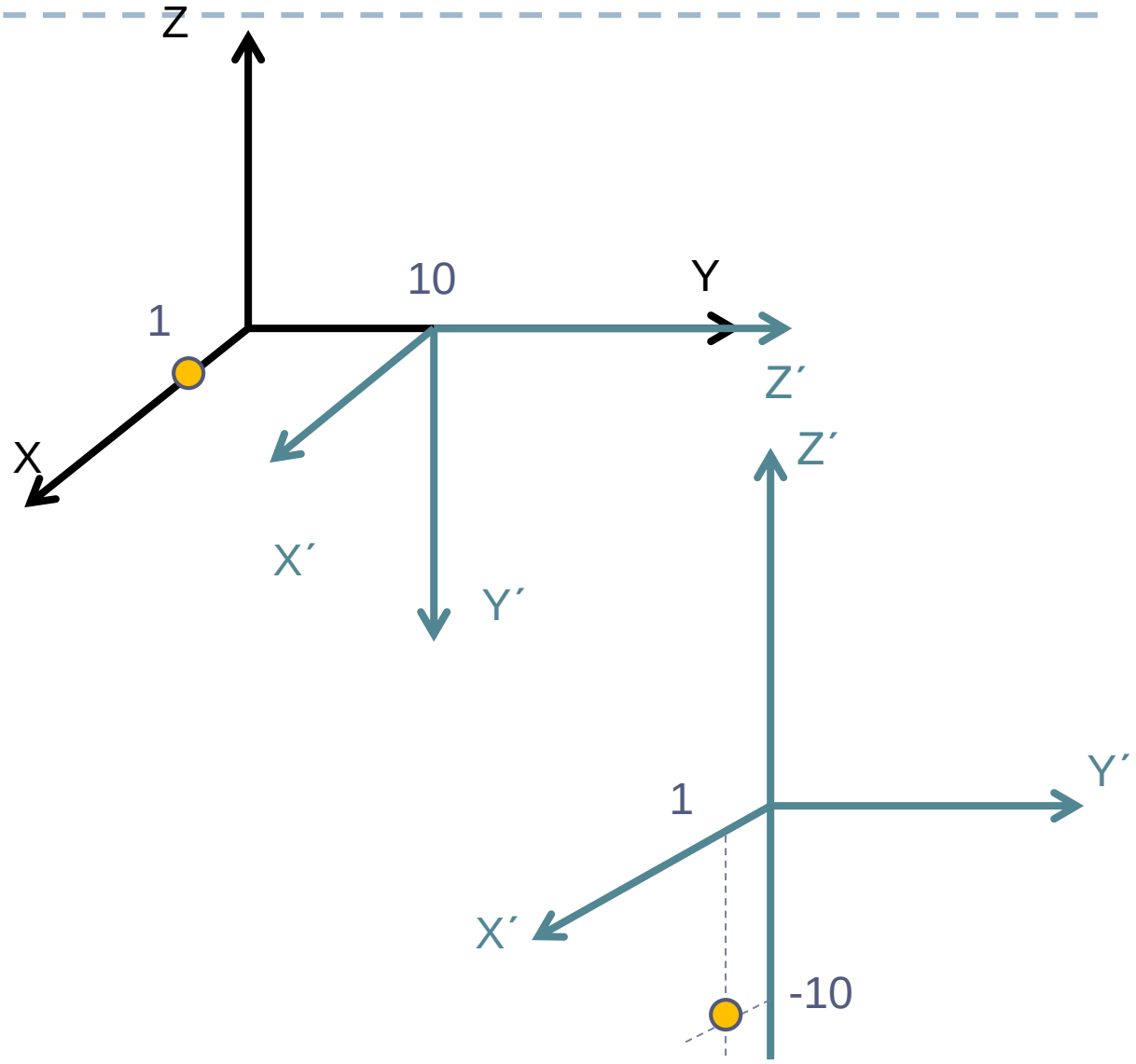
$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} x \\ -z \\ y-10 \\ w \end{bmatrix}$$



Translation Matrices

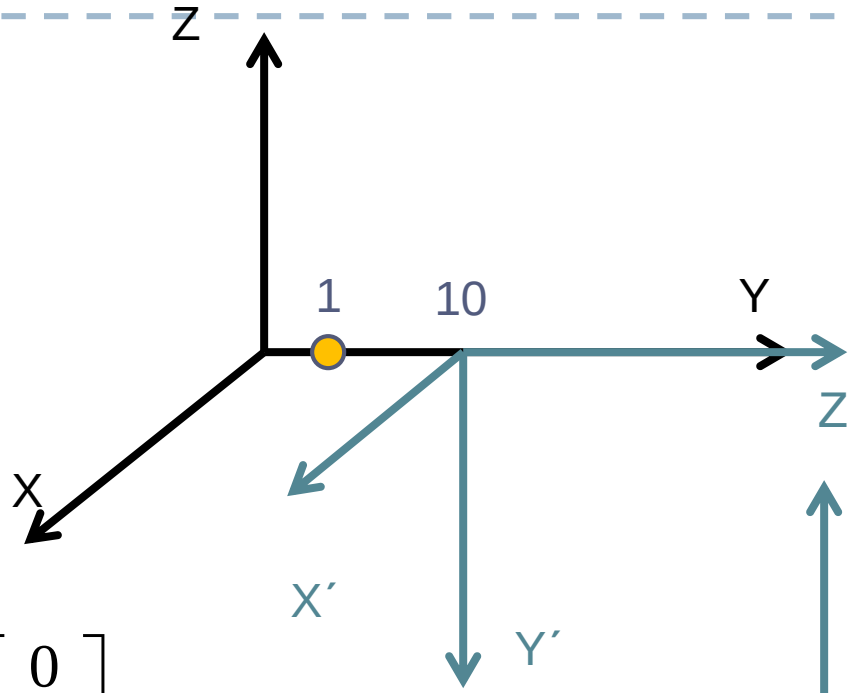
$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} x \\ -z \\ y-10 \\ w \end{bmatrix}$$

$$\begin{bmatrix} 1 \\ 0 \\ 0 \\ w \end{bmatrix} \rightarrow \begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ -10 \\ w \end{bmatrix}$$

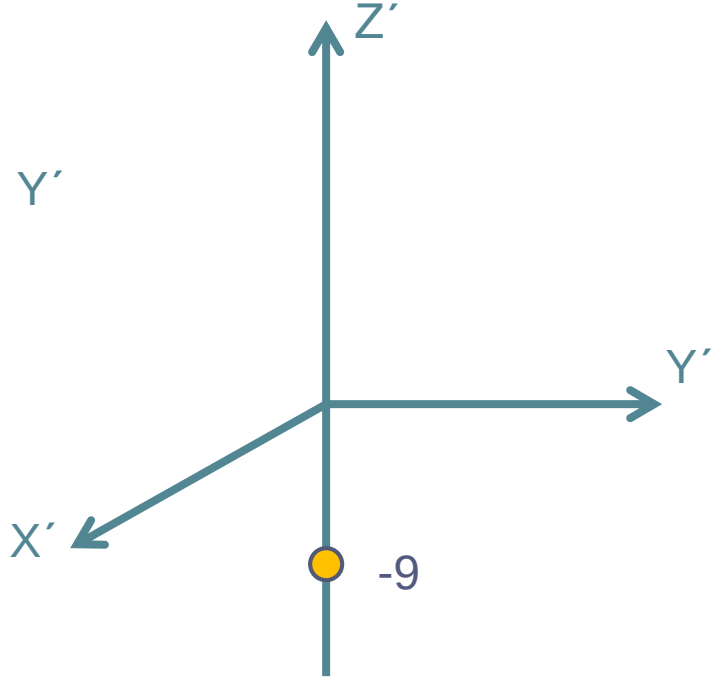


Translation Matrices

$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} x \\ -z \\ y-10 \\ w \end{bmatrix}$$



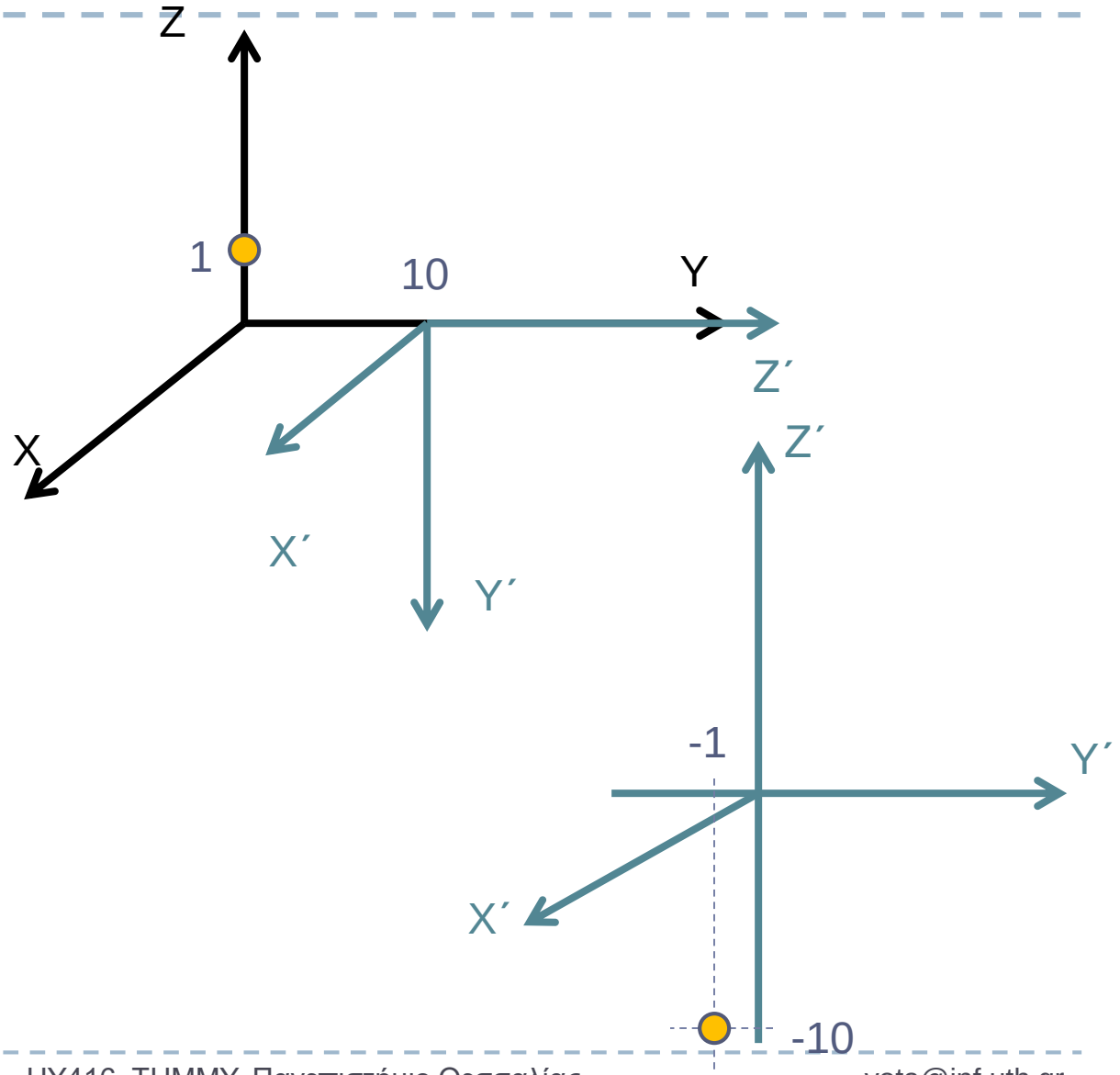
$$\begin{bmatrix} 0 \\ 1 \\ 0 \\ w \end{bmatrix} \rightarrow \begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1-10 \\ w \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -9 \\ w \end{bmatrix}$$



Translation Matrices

$$\begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} x \\ -z \\ y-10 \\ w \end{bmatrix}$$

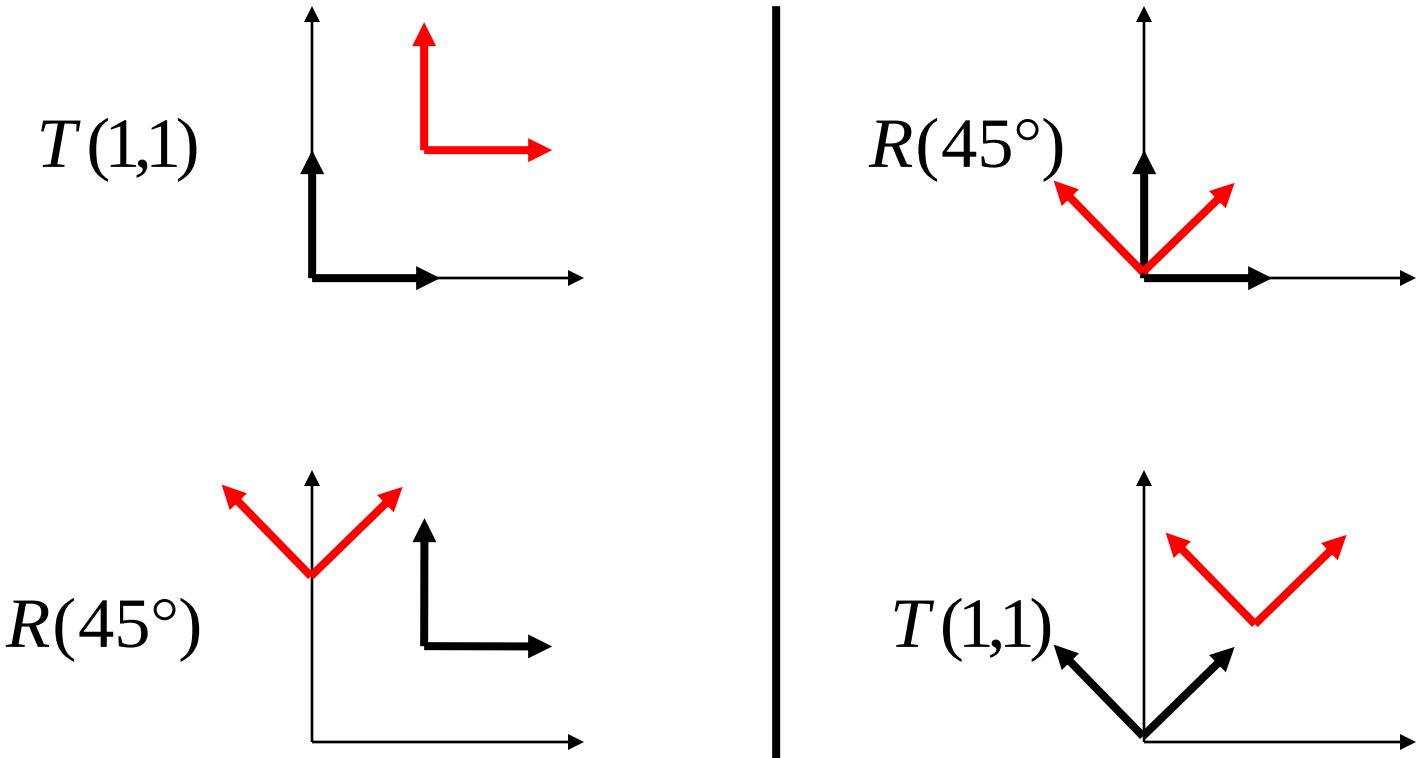
$$\begin{bmatrix} 0 \\ 0 \\ 1 \\ w \end{bmatrix} \rightarrow \begin{bmatrix} x' \\ y' \\ z' \\ w' \end{bmatrix} = \begin{bmatrix} 0 \\ -1 \\ -10 \\ w \end{bmatrix}$$



Transformation Commutativity

- *Is matrix multiplication, in general, commutative? Does* $\mathbf{AB} = \mathbf{BA}$*?*
- *What about rotation, scaling, and translation matrices?*
  - *Does* $\mathbf{R}_x\mathbf{R}_y = \mathbf{R}_y\mathbf{R}_x$*?*
  - *Does* $\mathbf{R}_A\mathbf{S} = \mathbf{S}\mathbf{R}_A$ *?*
  - *Does* $\mathbf{R}_A\mathbf{T} = \mathbf{T}\mathbf{R}_A$ *?*

Combining Translation & Rotation



Combining Transformations

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

$$\mathbf{v}'' = R\mathbf{v}' = RS\mathbf{v}$$

$$\mathbf{v}''' = T\mathbf{v}'' = TR\mathbf{v}' = TRS\mathbf{v}$$

$$\mathbf{v}''' = M\mathbf{v}$$

where $M = TRS$

3-D Rotation

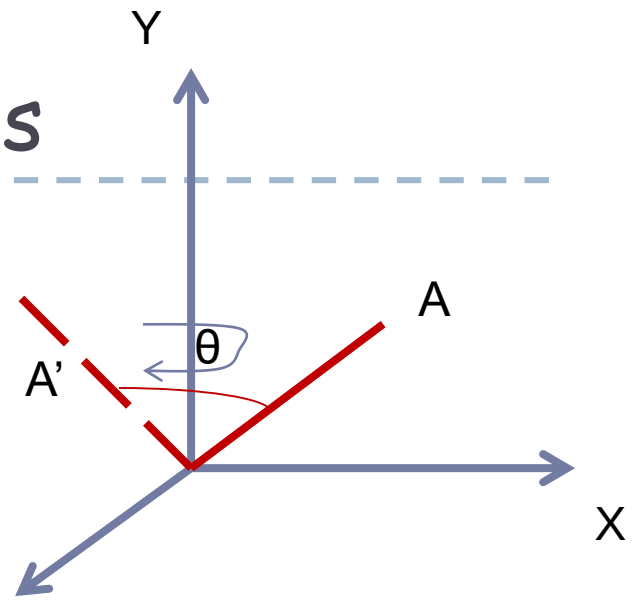
- General rotations in 3-D require rotating about an arbitrary axis of rotation
- Deriving the rotation matrix for such a rotation directly is a good exercise in linear algebra
- Another approach: express general rotation as composition of *canonical rotations*
  - Rotations about **X**, **Y**, **Z**

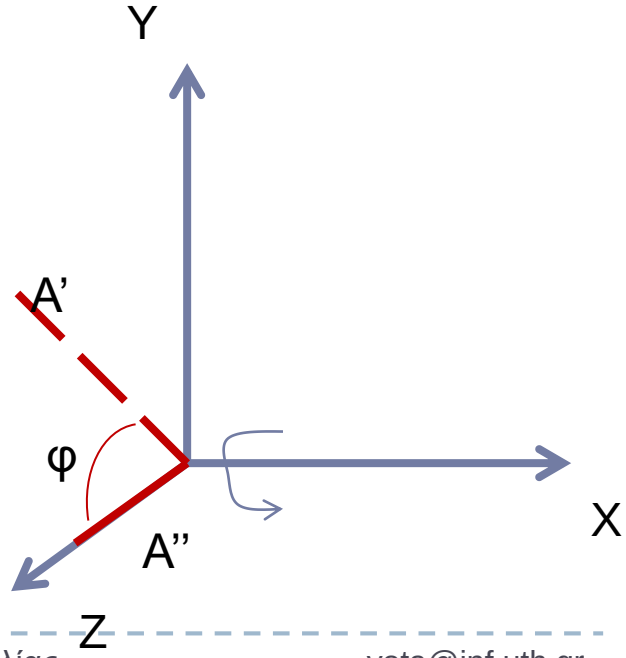
Composing Canonical Rotations

- Goal: rotate about arbitrary vector ***A*** by α
  - Idea: we know how to rotate about **X**,**Y**,**Z**
  - So, rotate about **Y** by θ until ***A*** lies in the YZ plane
  - Then rotate about **X** by φ until ***A*** coincides with **+Z**
  - Then rotate about **Z** by α
  - Then reverse the rotation about **X** (by $-\varphi$)
  - Then reverse the rotation about **Y** (by $-\theta$)

Composing Canonical Rotations

- First: rotating about **Y** by θ until **A** lies in YZ
  - Draw it…
- *How exactly do we calculate θ?*
  - Project **A** onto XZ plane
  - Find angle τ to **X**:
    θ = -(90° - τ) = τ - 90 °
- Second: rotating about **X** by φ until **A** lies on **Z**
- *How do we calculate φ?*





3-D Rotation Matrices

- So an arbitrary rotation about **A** composites several canonical rotations together
- We can express each rotation as a matrix
- Compositing transforms == multiplying matrices
- Thus we can express the final rotation as the product of canonical rotation matrices
- Thus we can express the final rotation with a single matrix!

Compositing Matrices

- So we have the following matrices:
  - ***p***: The point to be rotated about ***A*** by α
  - $R_{y\theta}$: Rotate about **Y** by θ
  - $R_{x\varphi}$: Rotate about **X** by φ
  - $R_{z\alpha}$: Rotate about **Z** by α
  - $R_{x\varphi}^{-1}$: Undo rotation about **X** by φ
  - $R_{y\theta}^{-1}$: Undo rotation about **Y** by θ
- *In what order should we multiply them?*

Compositing Matrices

- Remember: the transformations, in order, are written from *right* to *left*
  - In other words, the first matrix to affect the vector goes next to the vector, the second next to the first, etc.
  - This is the rule with column vectors (OpenGL); row vectors would be the opposite
- So in our case:

$$\mathbf{p}' = R_{y\theta}^{-1} R_{x\varphi}^{-1} R_{z\alpha} R_{x\varphi} R_{y\theta} \mathbf{p}$$

Rotation Matrices

- Notice these two matrices:
  $R_{x\,\varphi}$: Rotate about **X** by φ
  $R_{x\,\varphi}^{-1}$: Undo rotation about **X** by φ
- *How can we calculate $R_{x\,\varphi}^{-1}$?*
  - Obvious answer: calculate $R_{x\,(-\varphi)}$
  - Clever answer: exploit fact that rotation matrices are *orthonormal*
- *What is an orthonormal matrix?*
- *What property are we talking about?*

Rotation Matrices

- Rotation matrix is *orthogonal*
  - Columns/rows linearly independent
  - Columns/rows sum to 1
- The inverse of an orthogonal matrix is just its transpose:

$$\begin{bmatrix} a & b & c \\ d & e & f \\ h & i & j \end{bmatrix}^{-1} = \begin{bmatrix} a & b & c \\ d & e & f \\ h & i & j \end{bmatrix}^{T} = \begin{bmatrix} a & d & h \\ b & e & i \\ c & f & j \end{bmatrix}$$

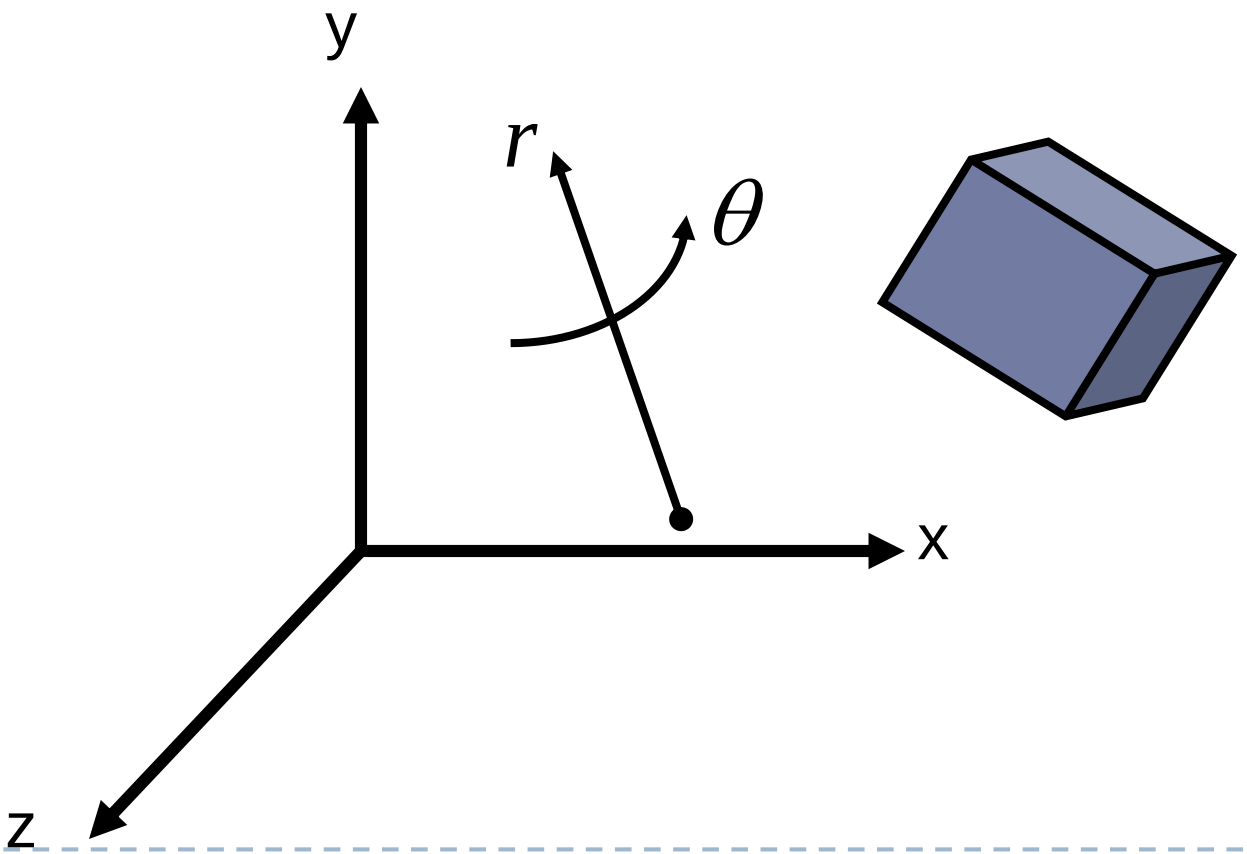
Rotation Matrix for Any Axis

- Given glRotated (angle, x, y, z)
  - Let c = cos(angle)
  - Let s = sin(angle)
  - And normalize the vector so that ||(x,y,z)|| == 1
- The produced matrix to rotate something by angle degrees around the axis (x,y,z) is:

$$\begin{bmatrix} xx(1-c)+c & xy(1-c)-zs & xz(1-c)+ys & 0 \\ yx(1-c)+zs & yy(1-c)+c & yz(1-c)-xs & 0 \\ zx(1-c)-ys & zy(1-c)+xs & zz(1-c)+c & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotations about an arbitrary axis

Rotate by θ around a unit axis r



An Alternative View

- We can view the rotation around an arbitrary axis as a set of simpler steps
- We know how to rotate and translate around the world coordinate system
- Can we use this knowledge to perform the rotation?

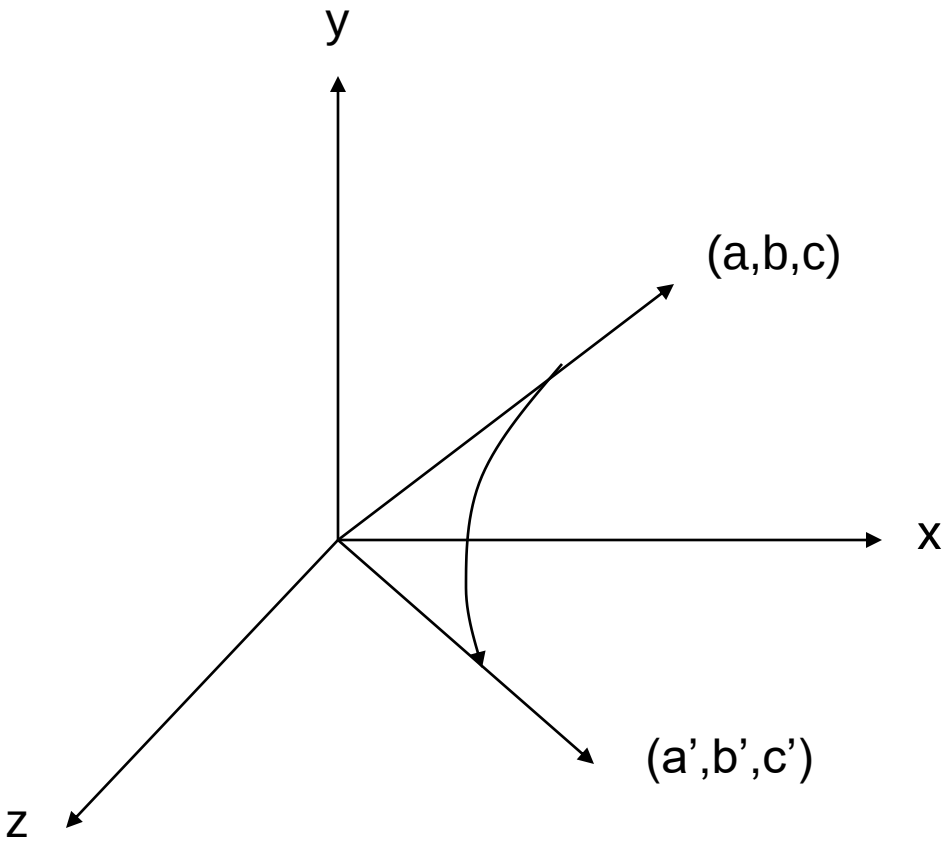
Rotation about an arbitrary axis

- Translate the space so that the origin of the unit vector is on the world origin
- Rotate such that the extremity of the vector now lies in the xz plane (x-axis rotation)
- Rotate such that the point lies in the z-axis (y-axis rotation)
- Perform the rotation around the z-axis
- Undo the previous transformations

Rotation about an arbitrary axis

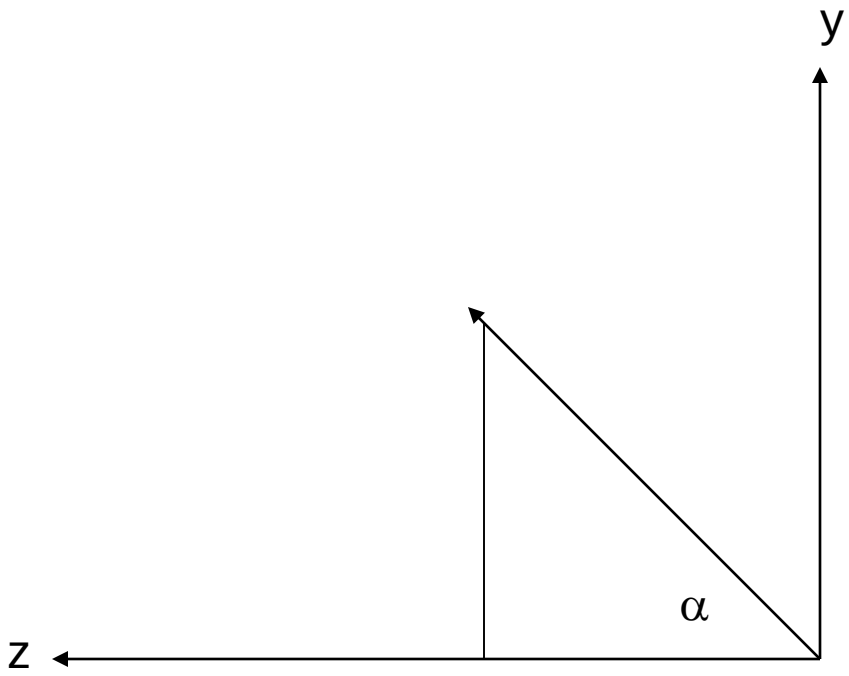
- Step 1

Rotate x-axis



Closer Look at Y-Z Plane

- Need to rotate α degrees around the x-axis



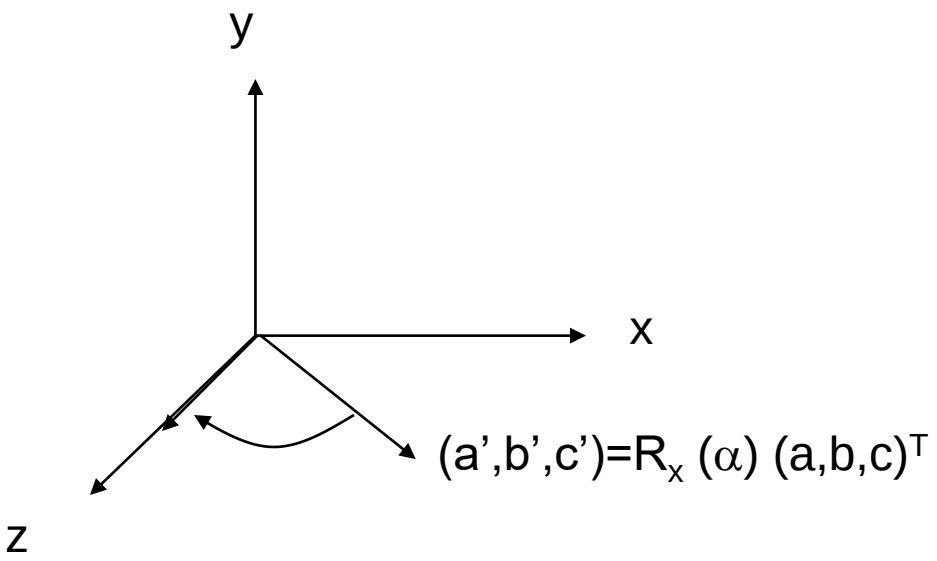
Equations for α

$$\sin(\alpha) = \frac{\| (0,0,1) \times (0,b,c) \|}{\| (0,0,1) \| \| (0,b,c) \|}$$

$$\cos(\alpha) = \frac{(0,b,c) \bullet (0,0,1)}{\| (0,b,c) \| \| (0,0,1) \|}$$

Rotation about the Y-axis

- Using the same analysis as before, we need to rotate β degrees around the Y-axis



Equations for β

$$\sin(\beta) = \frac{\| (0,0,1) \times (a',b',c') \|}{\| (0,0,1) \| \| (a',b',c') \|}$$

$$\cos(\beta) = \frac{(a',b',c') \bullet (0,0,1)}{\| (a',b',c') \| \| (0,0,1) \|}$$

Rotation about the Z-axis

- Now, it is aligned with the Z-axis, thus we can simply rotate θ degrees around the Z-axis.
- Then undo all the transformations we just did

Equation summary

$$rot_{axis}(\theta) = \begin{pmatrix} a' \\ b' \\ c' \end{pmatrix} = T^{-1}R_x^{-1}(\alpha)R_y^{-1}(\beta)R_z(\theta)R_y(\beta)R_x(\alpha)T \begin{pmatrix} a \\ b \\ c \end{pmatrix}$$

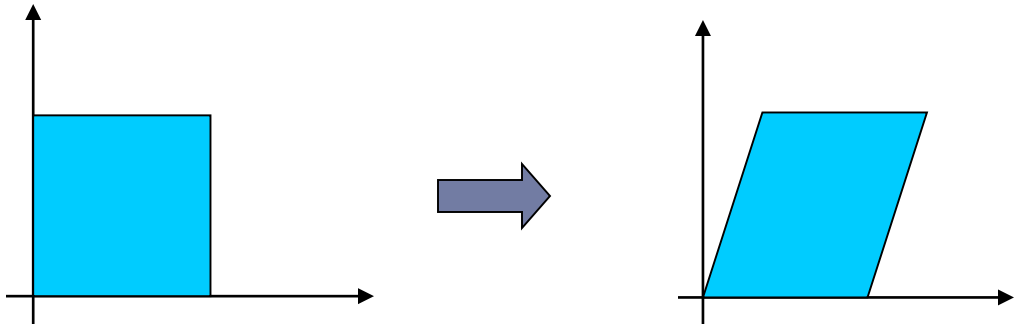
Other Transformations

Deformations

Transformations that do not preserve shape

- Non-uniform scaling
- Shearing
- Tapering
- Twisting
- Bending

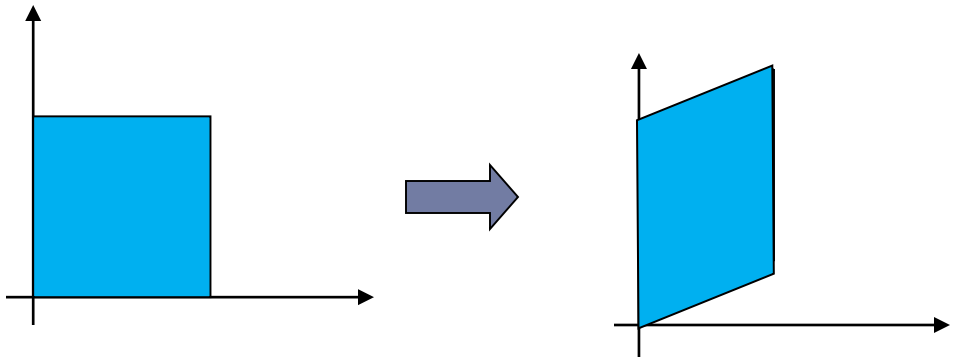
Shearing



- Y coordinates are unaffected, but x coordinates are translated linearly with y
- That is:
  - $y' = y$
  - $x' = x + y * h$

$$\begin{vmatrix} x \\ y \\ 1 \end{vmatrix} = \begin{vmatrix} 1 & h & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{vmatrix} * \begin{vmatrix} x \\ y \\ 1 \end{vmatrix}$$

Shearing in Y



$$\begin{vmatrix} x \\ y \\ 1 \end{vmatrix} = \begin{vmatrix} 1 & 0 & 0 \\ g & 1 & 0 \\ 0 & 0 & 1 \end{vmatrix} * \begin{vmatrix} x \\ y \\ 1 \end{vmatrix}$$

Interesting Facts:

- A 2D rotation is three shears
- Shearing will not change the area of the object
- Any 2D shearing can be done by a rotation, followed by a scaling, and followed by a rotation

Shearing

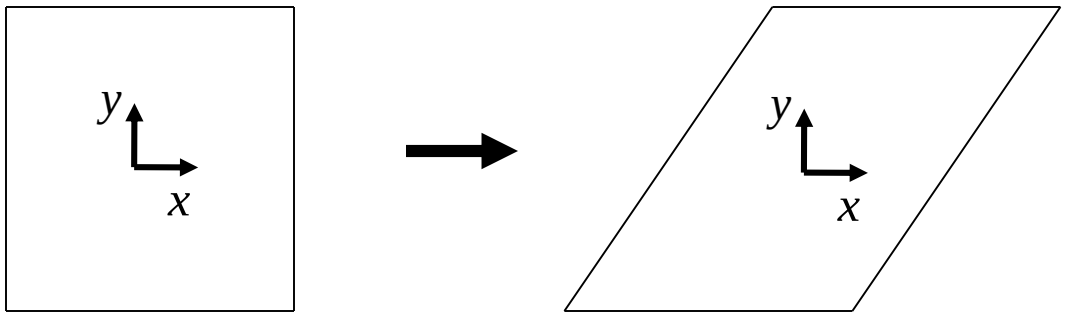
$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & s_{xy} & s_{xz} & 0 \\ s_{yx} & 1 & s_{yz} & 0 \\ s_{zx} & s_{zy} & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$s_{xy} = 1$

$s_{xz} = 0$

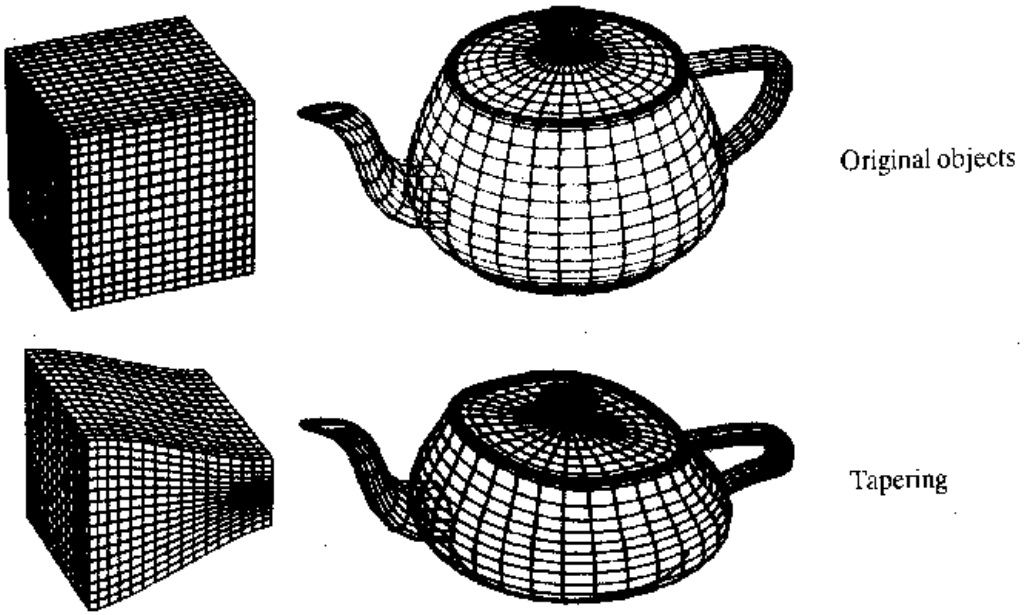
$s_{yx} = s_{yz} = 0$

$s_{zx} = s_{zy} = 0$



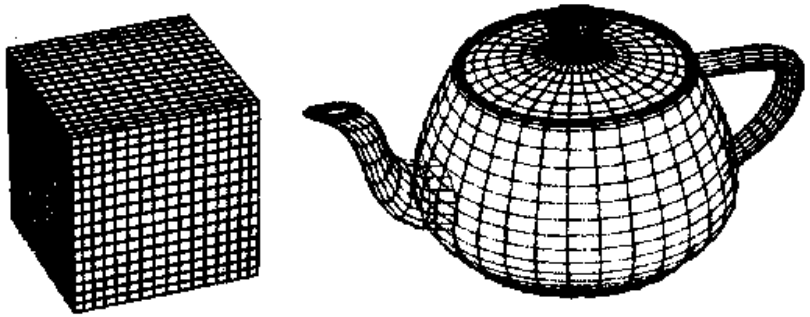
Tapering

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & f(x) & 0 & 0 \\ 0 & 0 & f(x) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

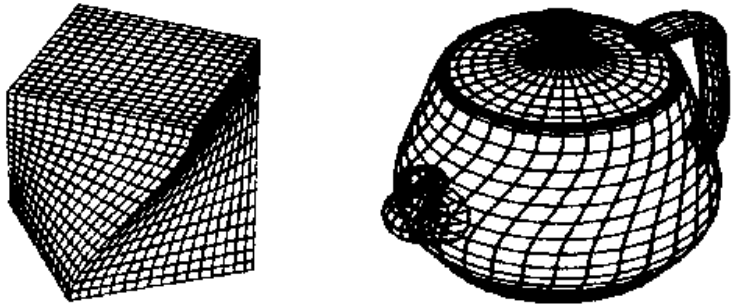


Twisting

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



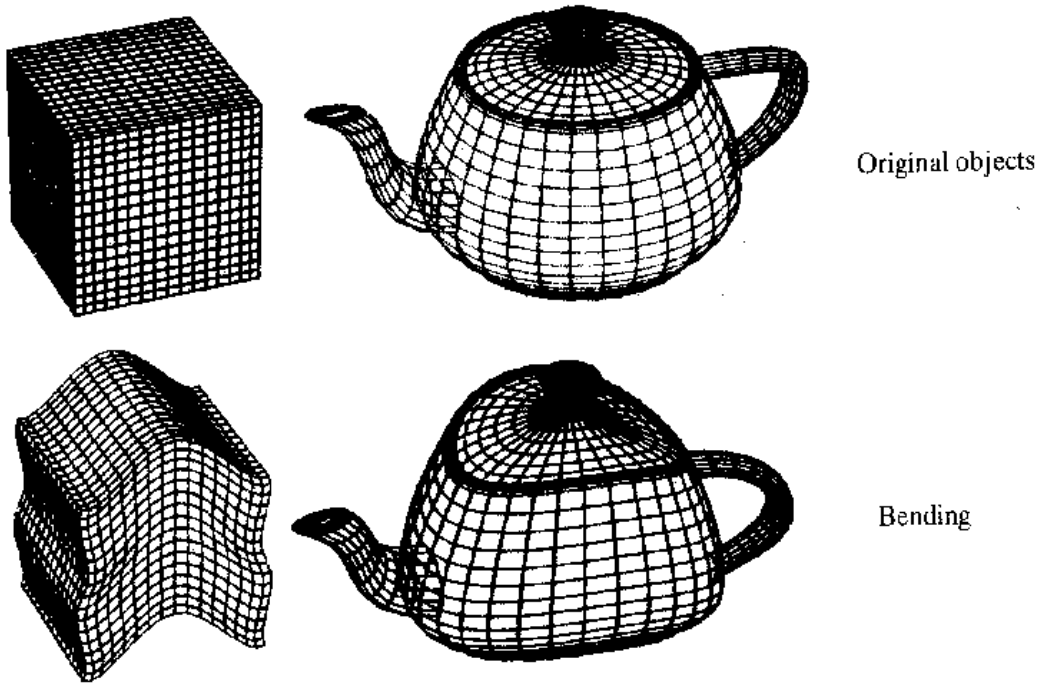
Original objects



Twisting

Bending

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & f(y) & g(y) & 0 \\ 0 & h(y) & k(y) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$



Original objects

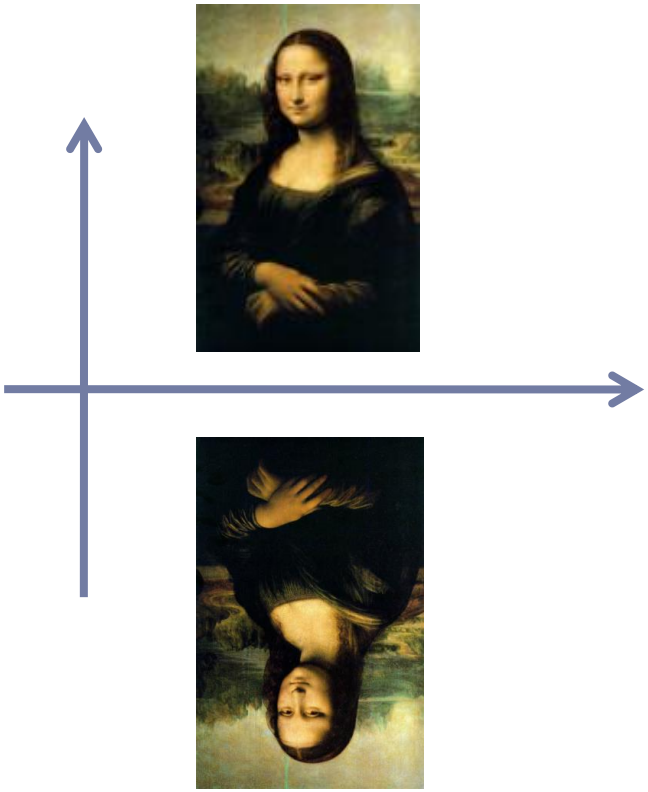
Bending

Reflection

Reflection

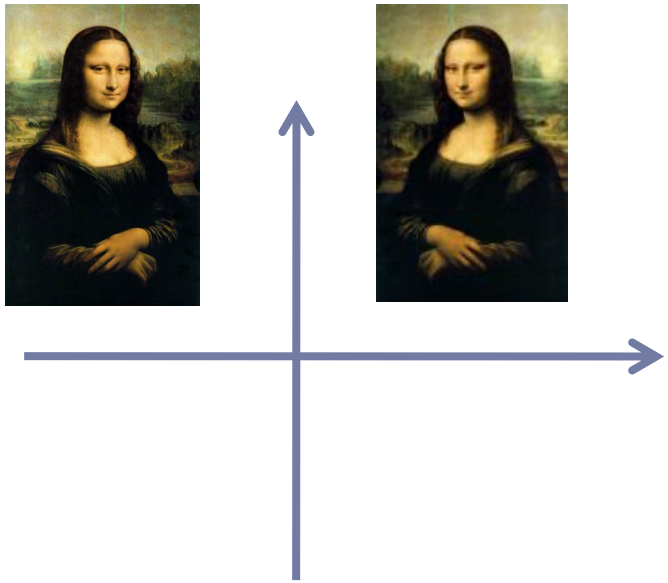
Reflection

Reflection about X-axis



$$\begin{vmatrix} x \\ y \\ 1 \end{vmatrix} = \begin{vmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{vmatrix} * \begin{vmatrix} x \\ y \\ 1 \end{vmatrix}$$

Reflection about Y-axis



$$\begin{vmatrix} x \\ y \\ 1 \end{vmatrix} = \begin{vmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{vmatrix} * \begin{vmatrix} x \\ y \\ 1 \end{vmatrix}$$

Affine Transformation

- Translation, Scaling, Rotation, Shearing are all affine transformation

Affine Transformation

- Translation, Scaling, Rotation, Shearing are all affine transformation
- Affine transformation – transformed point P' (x',y') is a linear combination of the original point P (x,y), i.e.

$$\begin{vmatrix} x' \\ y' \\ 1 \end{vmatrix} = \begin{vmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ 0 & 0 & 1 \end{vmatrix} \cdot \begin{vmatrix} x \\ y \\ 1 \end{vmatrix}$$

Affine Transformation

- Translation, Scaling, Rotation, Shearing (Στρέβλωση) are all affine transformation
- Affine transformation – transformed point P′ (x′,y′) is a linear combination of the original point P (x,y), i.e.

$$\begin{vmatrix} x' \\ y' \\ 1 \end{vmatrix} = \begin{vmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ 0 & 0 & 1 \end{vmatrix} \cdot \begin{vmatrix} x \\ y \\ 1 \end{vmatrix}$$

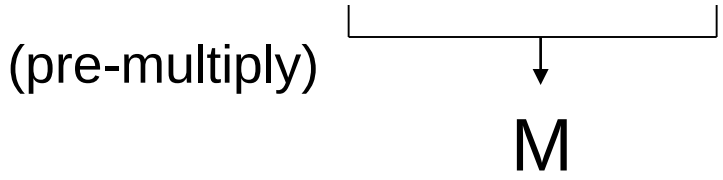
- Any 2D affine transformation can be decomposed into a rotation, followed by a scaling, followed by a shearing, and followed by a translation.

  Affine matrix = translation x shearing x scaling x rotation

Composing Transformation

- Composing Transformation – the process of applying several transformation in succession to form one overall transformation
- If we apply transform a point P using M1 matrix first, and then transform using M2, and then M3, then we have:

$$(M3 \times (M2 \times (M1 \times P))) = M3 \times M2 \times M1 \times P$$



Composing Transformation

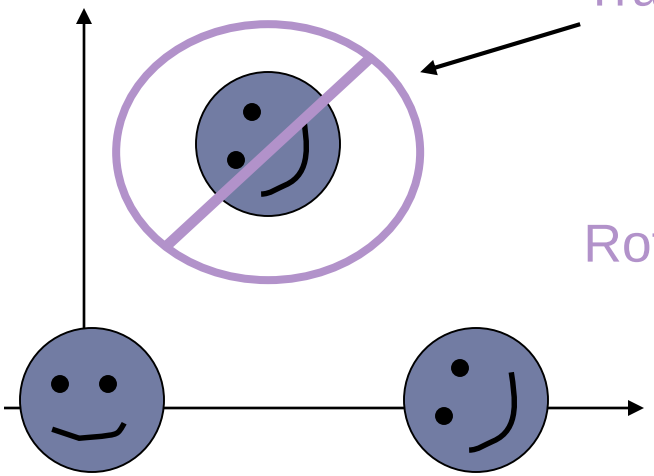
- Matrix multiplication is associative

  M3 x M2 x M1 = (M3 x M2) x M1 = M3 x (M2 x M1)
- Transformation products may not be commutative A x B != B x A
- Some cases where A x B = B x A

A	B
translation	translation
scaling	scaling
rotation	rotation
uniform scaling (sx = sy)	rotation

Transformation Order Matters!

- Example: rotation and translation are not commutative



Translate (5,0) and then Rotate 60 degree

OR

Rotate 60 degree and then translate (5,0)??

Rotate and then translate !!

Finding Affine Transformations

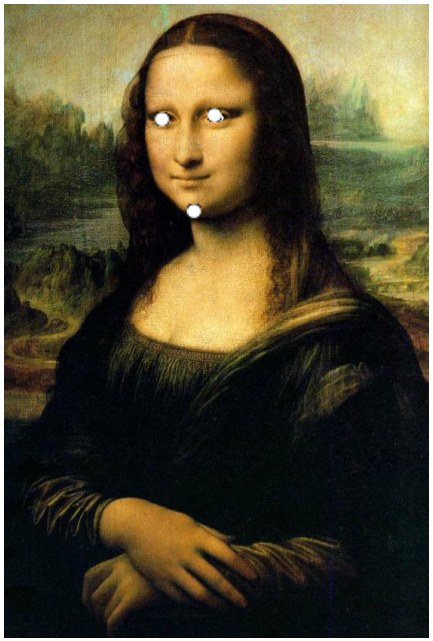
- Image of 3 points determines affine transformation

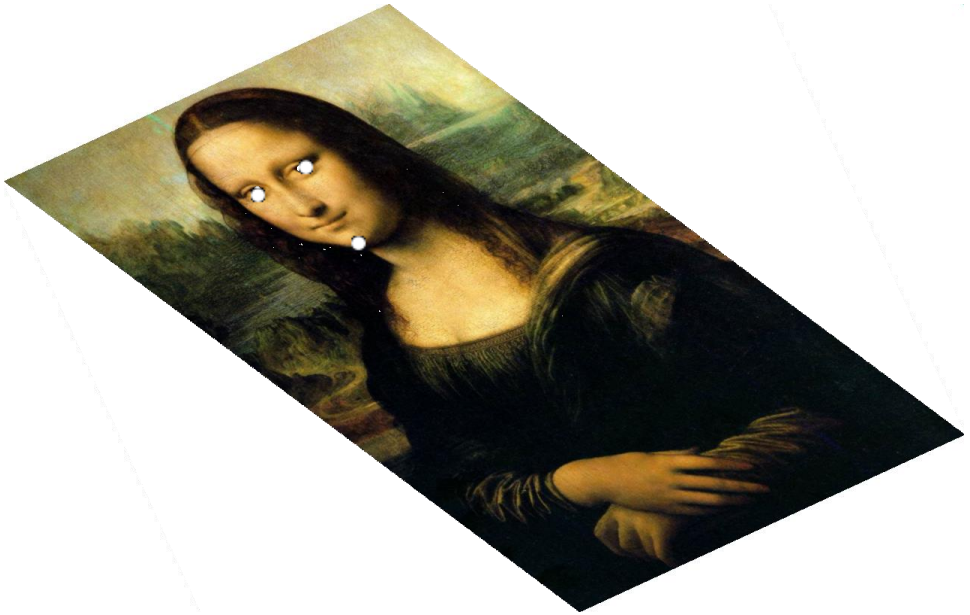
Finding Affine Transformations

- Image of 3 points determines affine transformation

Finding Affine Transformations

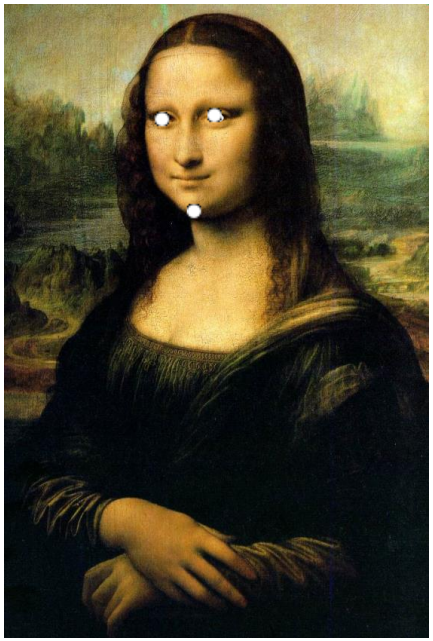
- Image of 3 points determines affine transformation

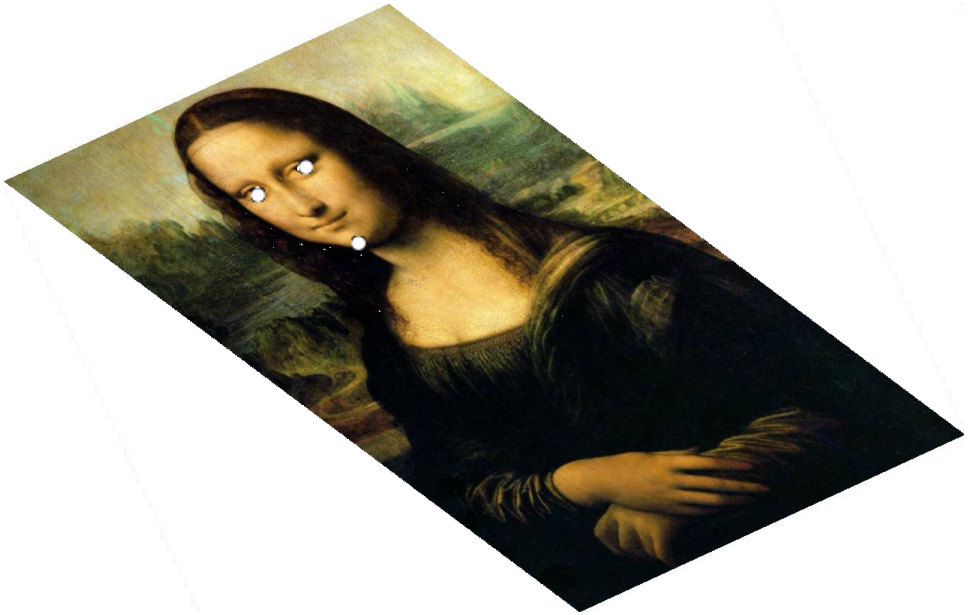




Finding Affine Transformations

- Image of 3 points determines affine transformation





$$\begin{pmatrix} M_{11} & M_{12} & M_{13} \\ M_{21} & M_{22} & M_{23} \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} p_x & q_x & r_x \\ p_y & q_y & r_y \\ 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} \hat{p}_x & \hat{q}_x & \hat{r}_x \\ \hat{p}_y & \hat{q}_y & \hat{r}_y \\ 1 & 1 & 1 \end{pmatrix}$$