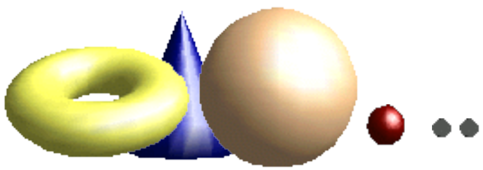
CENG 477
Introduction to Computer Graphics

Modeling Transformations

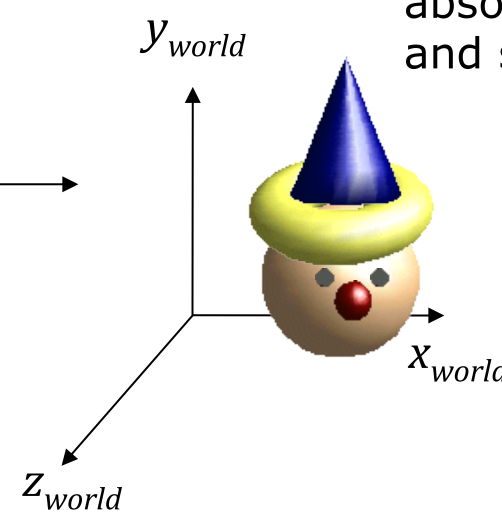
Modeling Transformations

- Model coordinates to World coordinates:



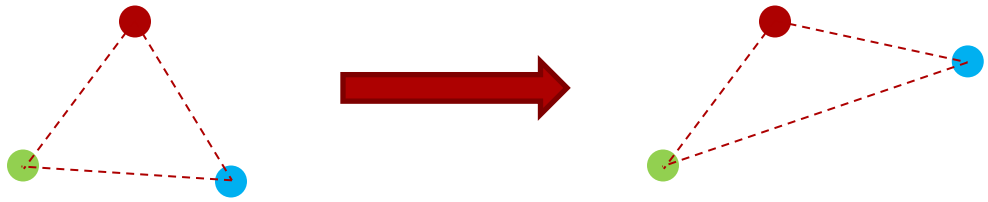
Model coordinates:
All shapes with their
local coordinates
and sizes.

World coordinates:
All shapes with their
absolute coordinates
and sizes.



Basic Geometric Transformations

- Used for modeling, animation as well as viewing
- What to transform?
  - We typically transform the vertices (points) and vectors describing the shape (such as the surface normal)

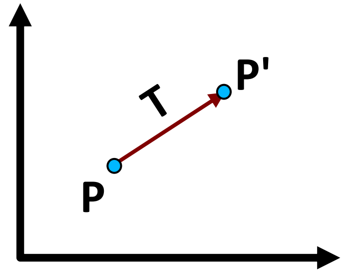


- This works due to the linearity of transformations
- Some, but not all, transformations may preserve attributes like sizes, angles, ratios of the shape

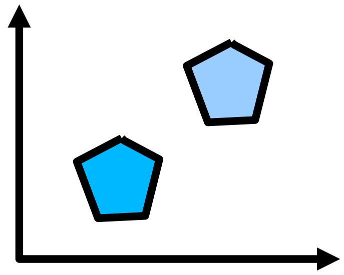
Translation

- Simply move the object to a relative position

$$x' = x + t_x \quad y' = y + t_y$$



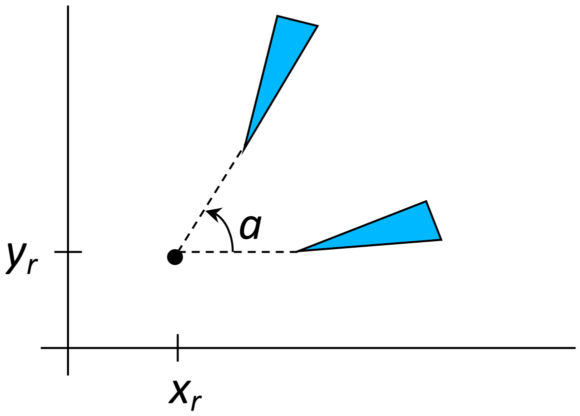
$$\mathbf{P} = \begin{bmatrix} x \\ y \end{bmatrix} \quad \mathbf{T} = \begin{bmatrix} t_x \\ t_y \end{bmatrix} \quad \mathbf{P}' = \begin{bmatrix} x' \\ y' \end{bmatrix}$$



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

Rotation

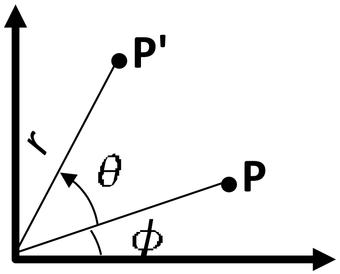
- A rotation is defined by a rotation axis and a rotation angle
- For 2D rotation, the parameters are rotation angle (θ) and the rotation point (x_r, y_r)
- We reposition the object in a circular path around the rotation point (pivot point)



Rotation

- When (x_r, y_r)=$(0,0)$ we have:

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



The original coordinates are:

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

Substituting them in the first equation we get:

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

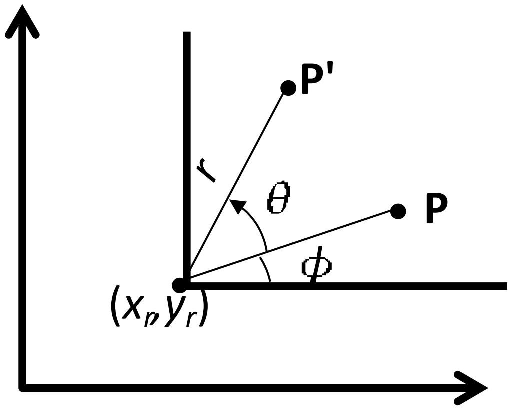
In the matrix form we have:

$$\mathbf{P}' = \mathbf{R}\cdot\mathbf{P}$$
$$\mathbf{R} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

Rotation

- Rotation around an arbitrary point (x_r, y_r)

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



- These equations can be written as matrix operations (we will see when we discuss homogeneous coordinates)

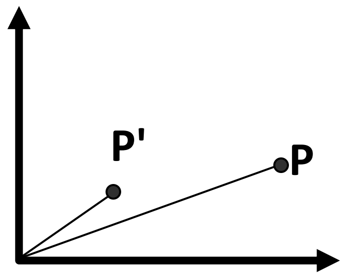
Scaling

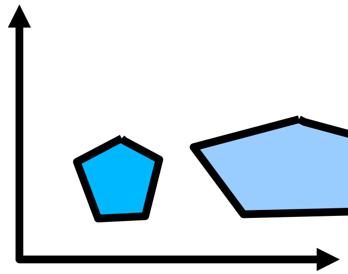
- Changes the size of an object
- Input: scaling factors (s_x, s_y)

$$x' = xs_x \qquad y' = ys_y$$

$$\mathbf{S} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix}$$

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





non-uniform vs.
uniform scaling

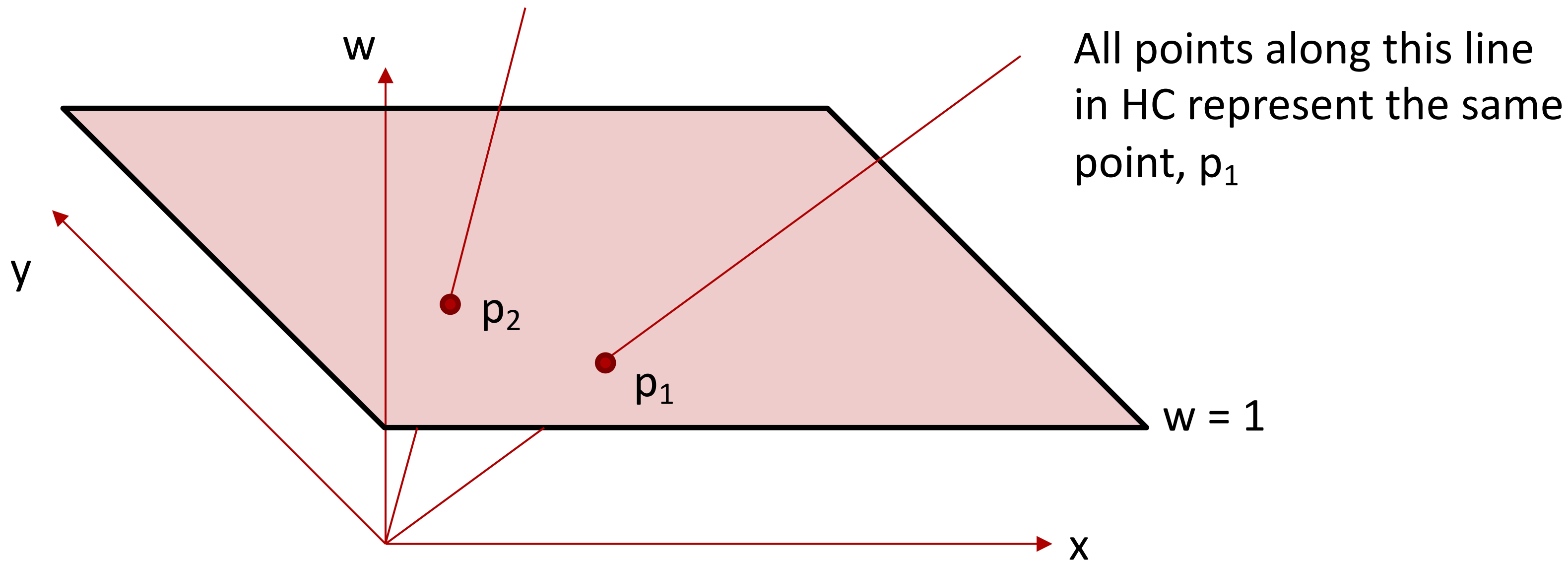
Homogenous Coordinates

- Translation is additive, rotation and scaling is multiplicative (and additive if you rotate around an arbitrary point or scale around a fixed point)
- **Goal:** Make all transformations as matrix operations
- **Solution:** Add a third dimension

$$x = \frac{x_h}{h} \quad y = \frac{y_h}{h} \quad P = \begin{bmatrix} x_h \\ y_h \\ h \end{bmatrix} = \begin{bmatrix} h \cdot x \\ h \cdot y \\ h \end{bmatrix} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Homogenous Coordinates

- In HC, each point now becomes a line
- The entire line represents the same point
- The original (non-homogeneous) point resides on the w=1 plane



Transformations in HC

- Translation: $$P' = T(t_x, t_y) \cdot P \quad \text{where} \quad T(t_x, t_y) = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

- Rotation: $$P' = R(\theta) \cdot P \quad \text{where} \quad R(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- Scaling: $$P' = S(s_x, s_y) \cdot P \quad \text{where} \quad S(s_x, s_y) = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Transforming Vectors

- Vectors can be rotated and scaled
- But translating a vector does not change it! Why?
  - A vector is a difference between two points
  - These two points translate the same way
  - So the vector remains the same
- Mathematically this can be achieved by setting the last coordinate of a *vector* to 0 (the last coordinate of points should be 1)

2D point $\begin{bmatrix} x \\ y \end{bmatrix}$ in HC is equal to $\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$ 2D vector $\begin{bmatrix} x \\ y \end{bmatrix}$ in HC is equal to $\begin{bmatrix} x \\ y \\ 0 \end{bmatrix}$

Composite Transformations

- Often, objects are transformed multiple times
- Such transformations can be combined into a single composite transformation
- E.g. Application of a sequence of transformations to a point:

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

Composite Transformations

- Composition of the same types of transformations is simple
- E.g. translation:

$$\mathbf{P}' = \mathbf{T}(t_{2x}, t_{2y}) \cdot \{\mathbf{T}(t_{1x}, t_{1y}) \cdot \mathbf{P}\}$$
$$= \{\mathbf{T}(t_{2x}, t_{2y}) \cdot \mathbf{T}(t_{1x}, t_{1y})\} \cdot \mathbf{P}$$

$$T(t_{2x}, t_{2y}) \cdot T(t_{1x}, t_{1y}) = \begin{bmatrix} 1 & 0 & t_{2x} \\ 0 & 1 & t_{2y} \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & t_{1x} \\ 0 & 1 & t_{1y} \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_{1x} + t_{2x} \\ 0 & 1 & t_{1y} + t_{2y} \\ 0 & 0 & 1 \end{bmatrix} = T(t_{1x} + t_{2x}, t_{1y} + t_{2y})$$

Composite Transformations

- Rotation and scaling are similar:

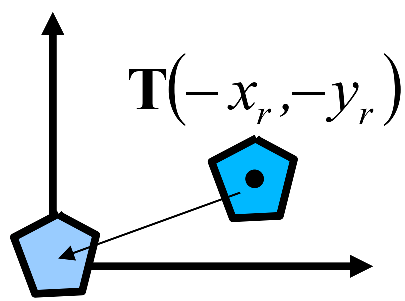
$$\mathbf{R}(\theta)\cdot\mathbf{R}(\varphi)=\begin{bmatrix}\cos\theta & -\sin\theta & 0\\ \sin\theta & \cos\theta & 0\\ 0 & 0 & 1\end{bmatrix}\cdot\begin{bmatrix}\cos\varphi & -\sin\varphi & 0\\ \sin\varphi & \cos\varphi & 0\\ 0 & 0 & 1\end{bmatrix}=$$

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

$$\mathbf{S}(s_{2x},s_{2y})\cdot\mathbf{S}(s_{1x},s_{1y})=\begin{bmatrix}s_{2x} & 0 & 0\\ 0 & s_{2y} & 0\\ 0 & 0 & 1\end{bmatrix}\cdot\begin{bmatrix}s_{1x} & 0 & 0\\ 0 & s_{1y} & 0\\ 0 & 0 & 1\end{bmatrix}=\begin{bmatrix}s_{1x}\cdot s_{2x} & 0 & 0\\ 0 & s_{1y}\cdot s_{2y} & 0\\ 0 & 0 & 1\end{bmatrix}=\mathbf{S}(s_{1x}\cdot s_{2x},s_{1y}\cdot s_{2y})$$

Rotation Around a Pivot Point

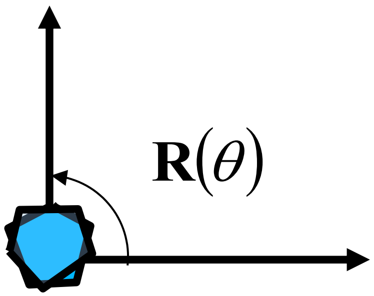
- **Step 1:** Translate the object so that the pivot point moves to the origin



$$M_1 = \mathbf{T}(-x_r, -y_r)$$

Rotation Around a Pivot Point

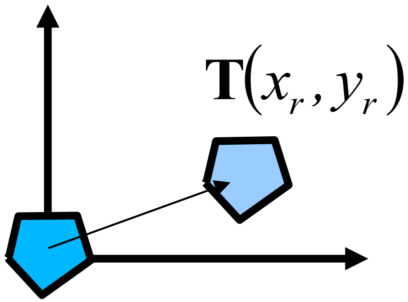
- **Step 2:** Rotate around origin



$$M_2 = \mathbf{R}(\theta)$$

Rotation Around a Pivot Point

- **Step 3:** Translate the object so that the pivot point is back to its original position



$$M_3 = \mathbf{T}(x_r, y_r)$$

Rotation Around a Pivot Point

- The composite transformation is equal to their successive application:

$$M = M_3 M_2 M_1 = \mathbf{T}(x_r, y_r)\mathbf{R}(\theta)\mathbf{T}(-x_r, -y_r)$$

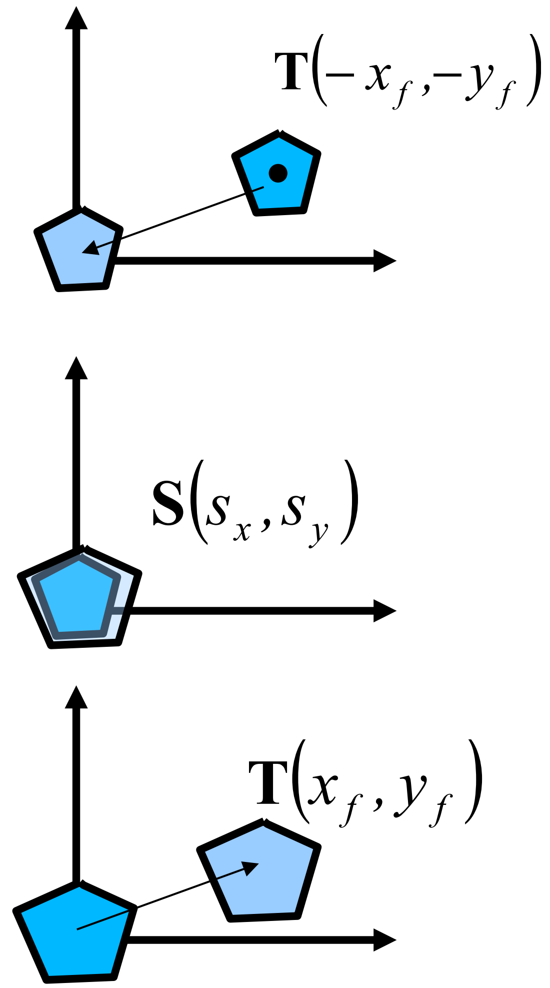
Scaling w.r.t. a Fixed Point

- The idea is the same:
  - Translate to origin
  - Scale
  - Translate back

$$\mathbf{T}(x_f, y_f)\cdot\mathbf{S}(s_x, s_y)\cdot\mathbf{T}(-x_f, -y_f) =$$

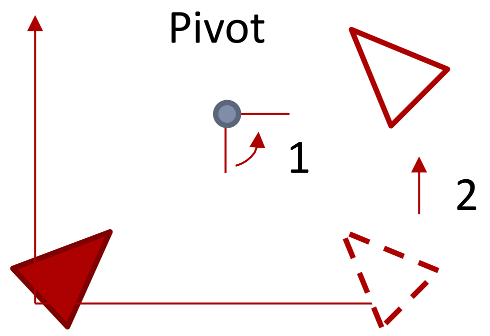
$$\begin{bmatrix} 1 & 0 & x_f \\ 0 & 1 & y_f \\ 0 & 0 & 1 \end{bmatrix}\cdot\begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}\cdot\begin{bmatrix} 1 & 0 & -x_f \\ 0 & 1 & -y_f \\ 0 & 0 & 1 \end{bmatrix} =$$

$$\begin{bmatrix} s_x & 0 & x_f(1-s_x) \\ 0 & s_y & y_f(1-s_y) \\ 0 & 0 & 1 \end{bmatrix}$$

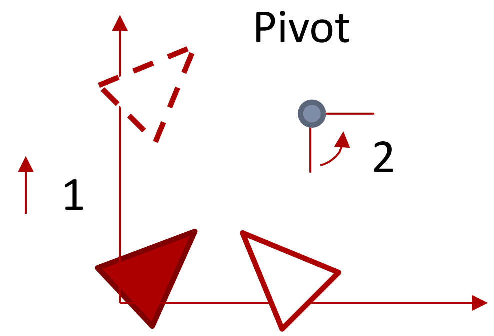


Order of matrix compositions

- Matrix composition is **not** commutative. So, be careful when applying a sequence of transformations.



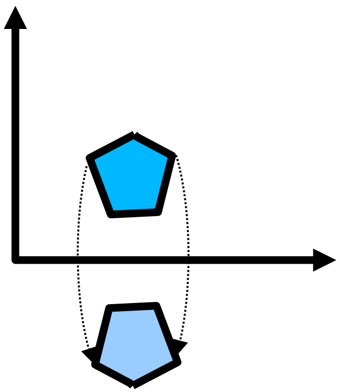
Rotation and translation



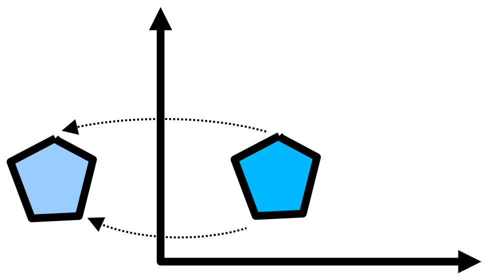
Translation and rotation

Other Transformations

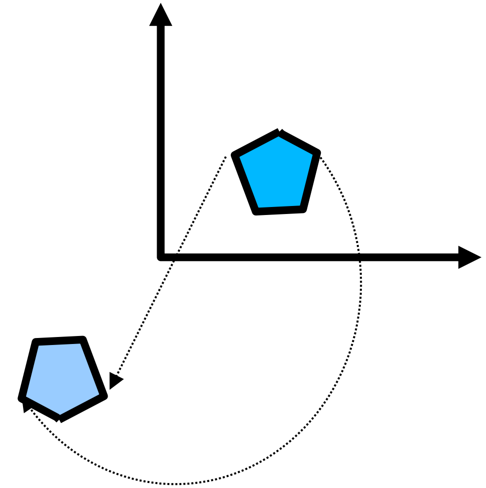
- **Reflection:** special case of scaling



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



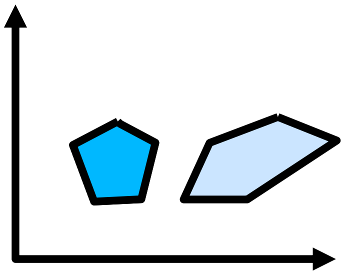
$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

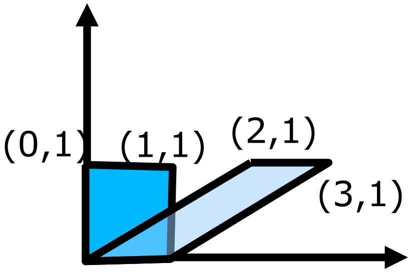


$$\begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Other Transformations

- **Shear:** Deform the shape like shifted slices (or deck of cards). Can be in **x** or **y** direction



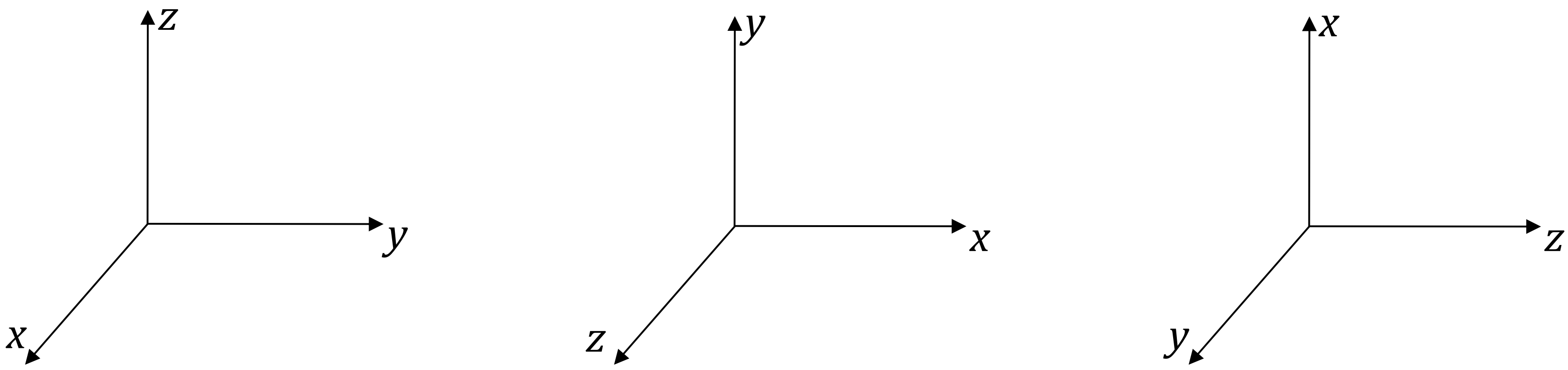


$$x' = x + sh_x \cdot y \qquad y' = y$$

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

3D Transformations

- Similar to 2D but with an extra z component
- We assume a right handed coordinate system
- With homogeneous coordinates we have 4 dimensions
- Basic transformations: Translation, rotation, scaling



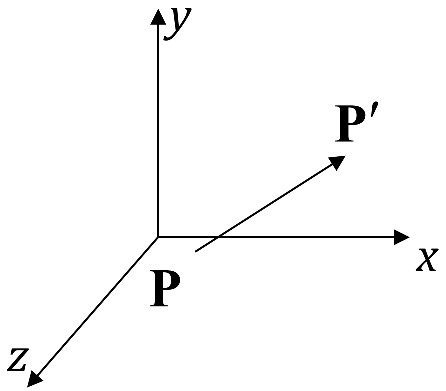
Equivalent ways of thinking about a right-handed CS

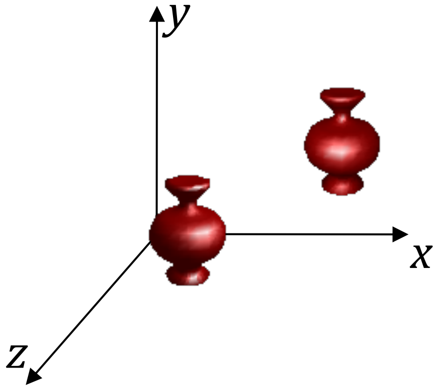
Translation

- Move the object by some offset:

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

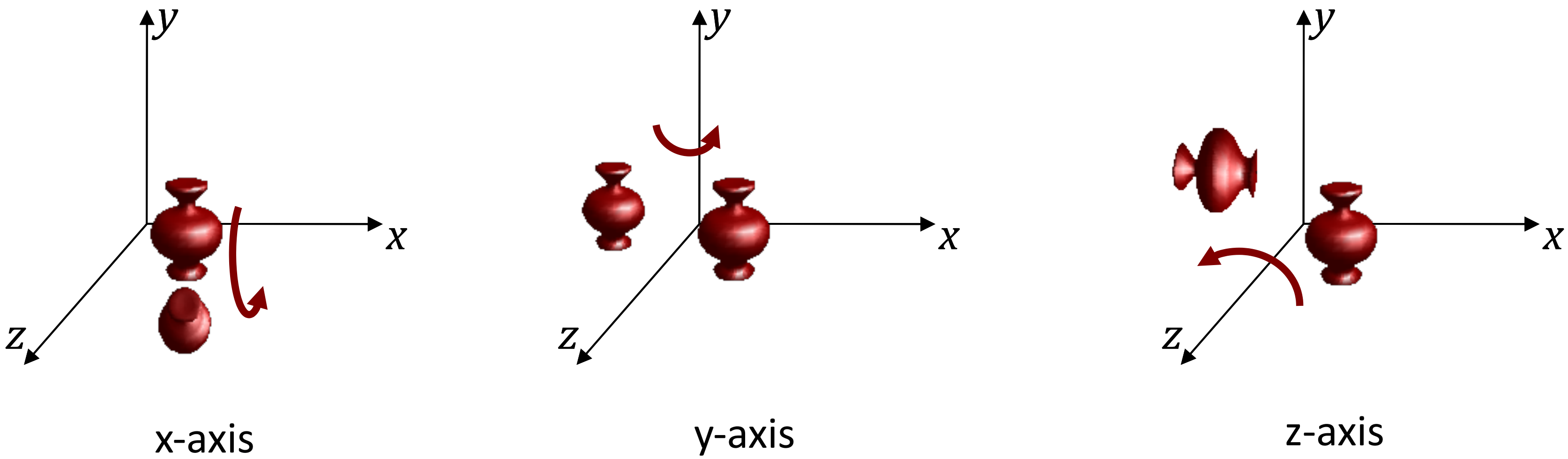
$$\mathbf{P}' = \mathbf{T} \cdot \mathbf{P}$$





Rotation

- Rotation around the coordinate axes



- Positive angles represent counter-clockwise (CCW) rotation when looking along the positive half towards origin

Rotation Around Major Axes

- Around x:

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

- Around y:

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

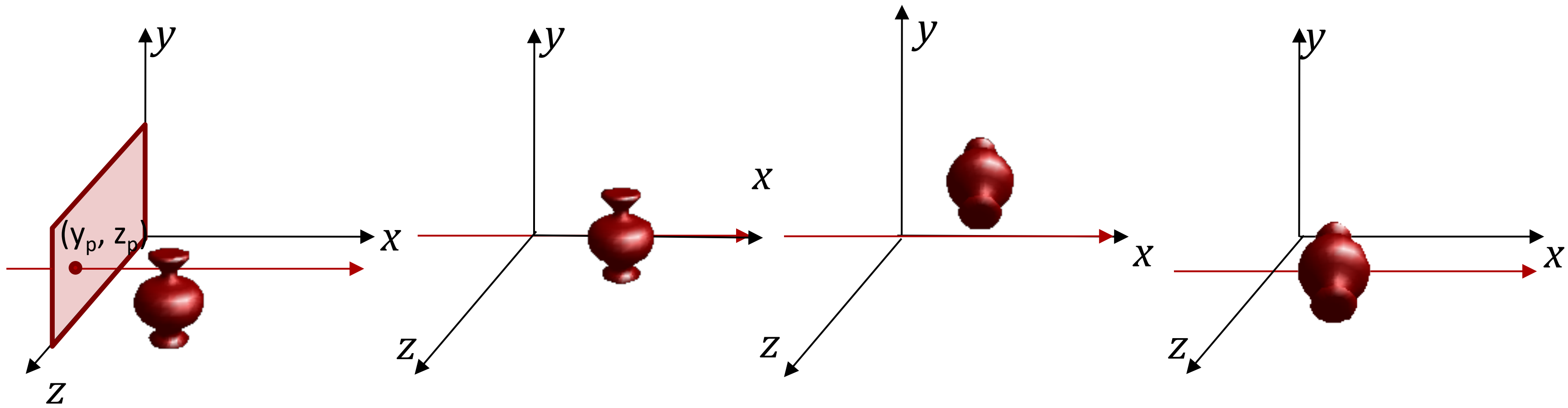
- Around z:

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

Rotation Around a Parallel Axis

- Rotating an object around a line parallel to one of the axes: Translate to a major axis, rotate, translate back
- **E.g.** rotate around a line parallel to x-axis:

$$\mathbf{P}' = \mathbf{T}(0, y_p, z_p) \cdot \mathbf{R}_x(\theta) \cdot \mathbf{T}(0, -y_p, -z_p) \cdot \mathbf{P}$$



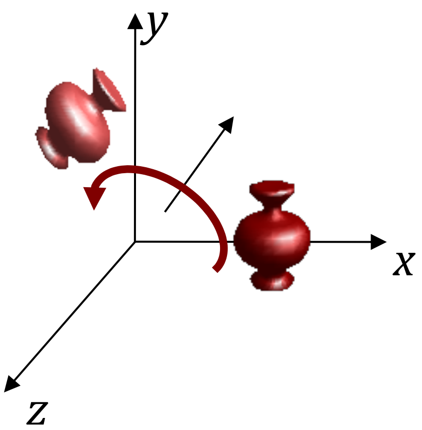
Translate

Rotate

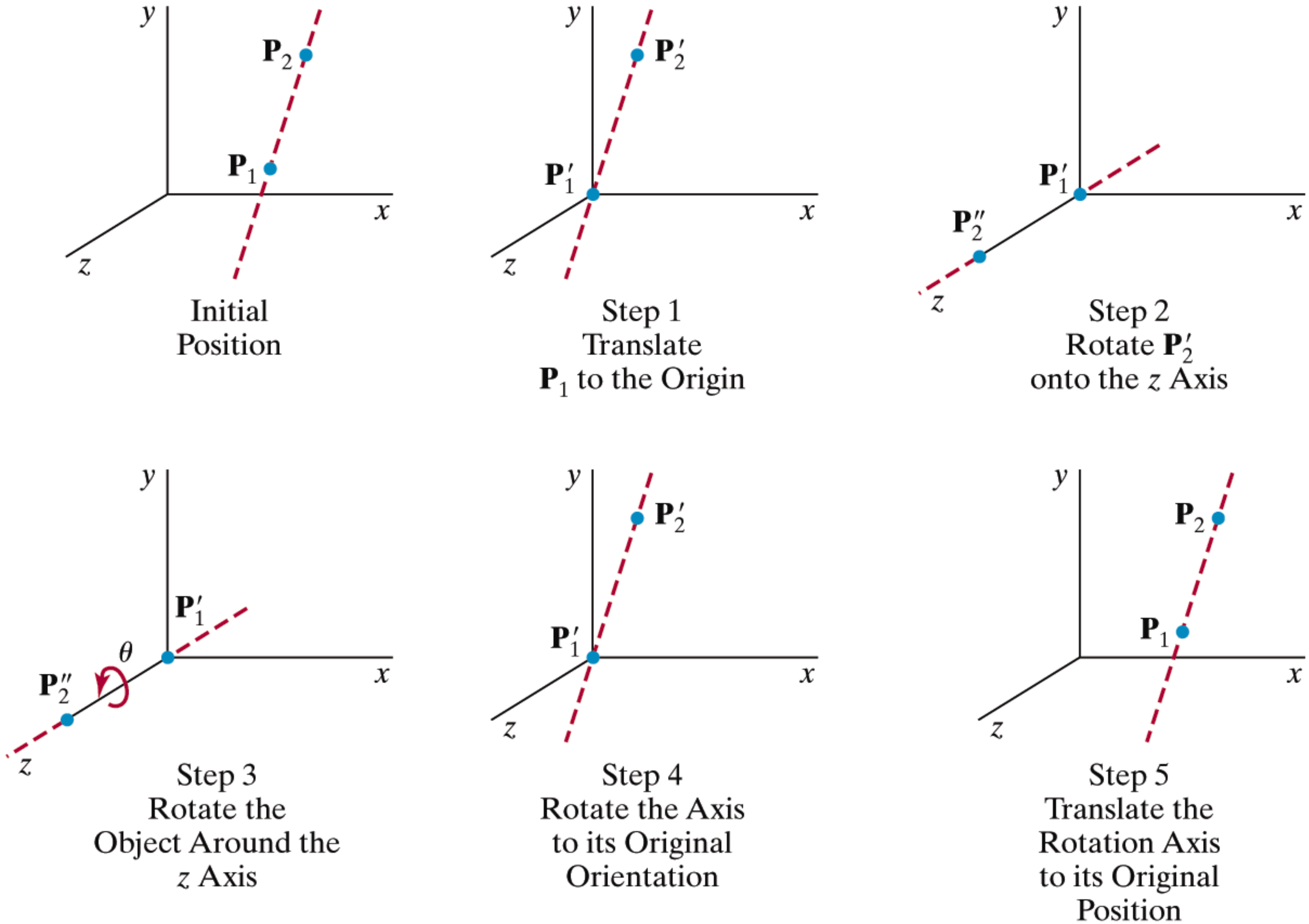
Translate back

Rotation Around an Arbitrary Axis

- **Step 1:** Translate the object so that the rotation axis passes though the origin
- **Step 2:** Rotate the object so that the rotation axis is aligned with one of the major axes
- **Step 3:** Make the specified rotation
- **Step 4:** Reverse the axis rotation
- **Step 5:** Translate back



Rotation Around an Arbitrary Axis



Rotation Around an Arbitrary Axis

- First determine the axis of rotation:

$$\mathbf{v} = \mathbf{P}_2 - \mathbf{P}_1 = (x_2 - x_1, y_2 - y_1, z_2 - z_1)$$

- **u** is the unit vector along **v**:

$$\mathbf{u} = \frac{\mathbf{v}}{|\mathbf{v}|} = (a, b, c)$$

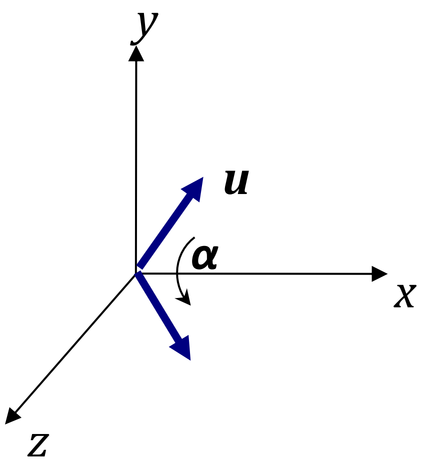
Rotation Around an Arbitrary Axis

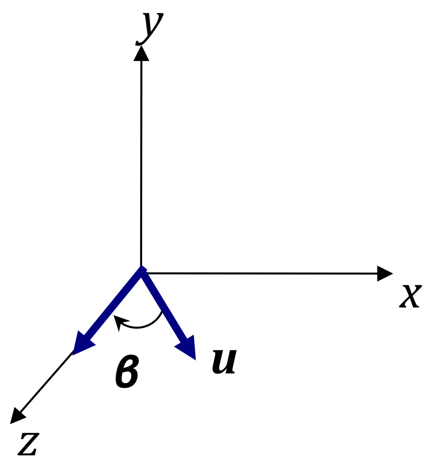
- Next translate $\mathbf{P_1}$ to origin:

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & -x_1 \\ 0 & 1 & 0 & -y_1 \\ 0 & 0 & 1 & -z_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotation Around an Arbitrary Axis

- Then align **u** with one of the major axis (**x**, **y**, or **z**)
- This is a two-step process:
  - Rotate around **x** to bring **u** onto **xz** plane (CCW)
  - Rotate around **y** to align the result with the **z**-axis (CW)

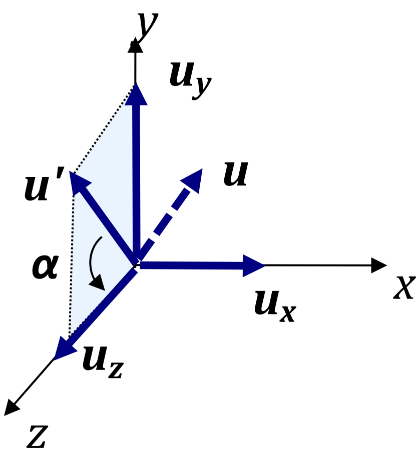




We need cosine and sine of angles α and ß

Rotation Around an Arbitrary Axis

- We need cosine and sine of angles α and ß:



$\mathbf{u} = u_x + u_y + u_z = u_x + \mathbf{u'}$

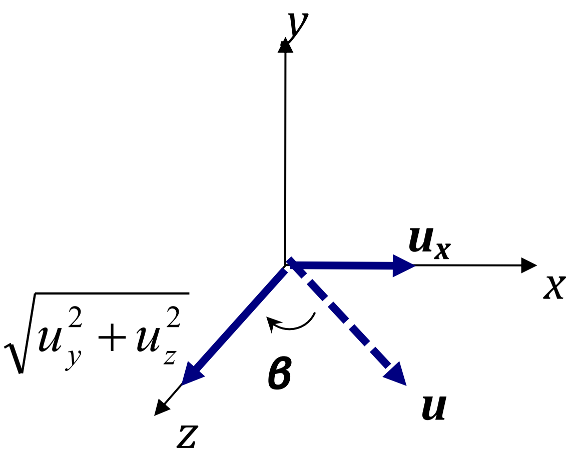
$$\cos\alpha = \frac{u_z}{|\mathbf{u}'|} = \frac{c}{d} \quad \text{where } d = \sqrt{b^2 + c^2}$$

$$\sin\alpha = \frac{u_y}{|\mathbf{u}'|} = \frac{b}{d}$$

$$\mathbf{R}_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{c}{d} & -\frac{b}{d} & 0 \\ 0 & \frac{b}{d} & \frac{c}{d} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Rotation Around an Arbitrary Axis

- We need cosine and sine of angles α and ß:



$$\mathbf{R}_y(\beta) = \begin{bmatrix} \frac{\sqrt{b^2+c^2}}{\sqrt{a^2+b^2+c^2}} & 0 & -\frac{a}{\sqrt{a^2+b^2+c^2}} & 0 \\ 0 & 1 & 0 & 0 \\ +\frac{a}{\sqrt{a^2+b^2+c^2}} & 0 & \frac{\sqrt{b^2+c^2}}{\sqrt{a^2+b^2+c^2}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\cos\beta = \frac{\sqrt{u_y^2+u_z^2}}{|\mathbf{u}|} = \frac{\sqrt{b^2+c^2}}{\sqrt{a^2+b^2+c^2}}$$

$$\sin\beta = \frac{u_x}{|\mathbf{u}|} = \frac{a}{\sqrt{a^2+b^2+c^2}}$$

Note that $\sqrt{a^2+b^2+c^2} = 1$

Rotation Around an Arbitrary Axis

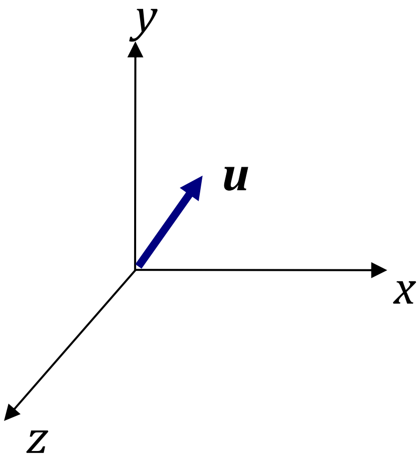
- Putting it all together:

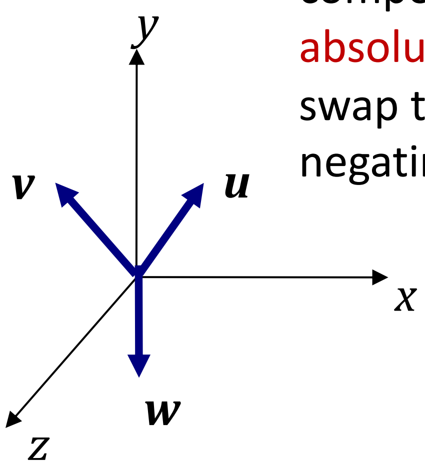
$$\mathbf{R}(\theta) = \mathbf{T}(x_1, y_1, z_1) \cdot \mathbf{R}_x(-\alpha) \cdot \mathbf{R}_y(+\beta) \cdot \mathbf{R}_z(\theta) \cdot \mathbf{R}_y(-\beta) \cdot \mathbf{R}_x(\alpha) \cdot \mathbf{T}(-x_1, -y_1, -z_1)$$

This is the actual desired rotation. Other terms are for alignment and undoing the alignment

Alternative Method

- Assume we want to rotate around the unit vector **u**:
- We create an orthonormal basis (ONB) **uvw**:





1) To find **v**, set the smallest component of **u** (in an absolute sense) to zero and swap the other two while negating one:

E.g. if **u** = (a, b, c) with c being the smallest absolute value then **v** = (-b, a, 0)

This corresponds to projecting the vector to the nearest major plane and rotating it 90° along the axis perpendicular to that plane

2) **w = u** x **v**

3) Normalize **v** and **w**

Note that we are just finding one of the infinitely many solutions

Alternative Method

- Now rotate **uvw** such that it aligns with **xyz**: call this transform M
- Rotate around **x** (**u** is now **x**)
- Undo the initial rotation: call this M^{-1}
- Finding M^{-1} (rotating **xyz** to **uvw**) is trivial:

- How to transform **x** = $[1\ 0\ 0\ 0]^T$ such that it turns into $[u_x\ u_y\ u_z\ 0]^T$
- Similar for the **y** and **z** axis

$$M^{-1} = \begin{bmatrix} u_x & v_x & w_x & 0 \\ u_y & v_y & w_y & 0 \\ u_z & v_z & w_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Verify that this matrix transforms **x** to **u**, **y** to **v**, and **z** to **w**

Alternative Method

- Finding M is also trivial as M^{-1} is an orthonormal matrix (all rows and columns are orthogonal unit vectors)
- For such matrices, inverse is equal to transpose:

$$M = \begin{bmatrix} u_x & u_y & u_z & 0 \\ v_x & v_y & v_z & 0 \\ w_x & w_y & w_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Alternative Method

- The final rotation transform is:

$$M^{-1}R_x(\theta)M$$

- We assumed that the origin of **uvw** is the same as the origin of **xyz**
- Otherwise, we should account for this difference:

$$T^{-1}M^{-1}R_x(\theta)MT$$

Undo the translation

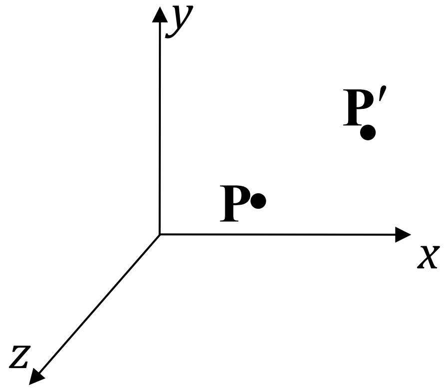
Translate the origin of uvw to xyz

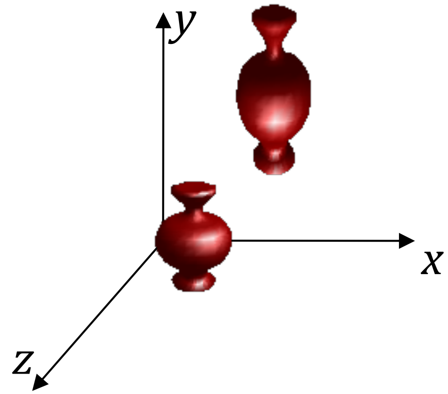
Scaling

- Change the coordinates of the object by scaling factors

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

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

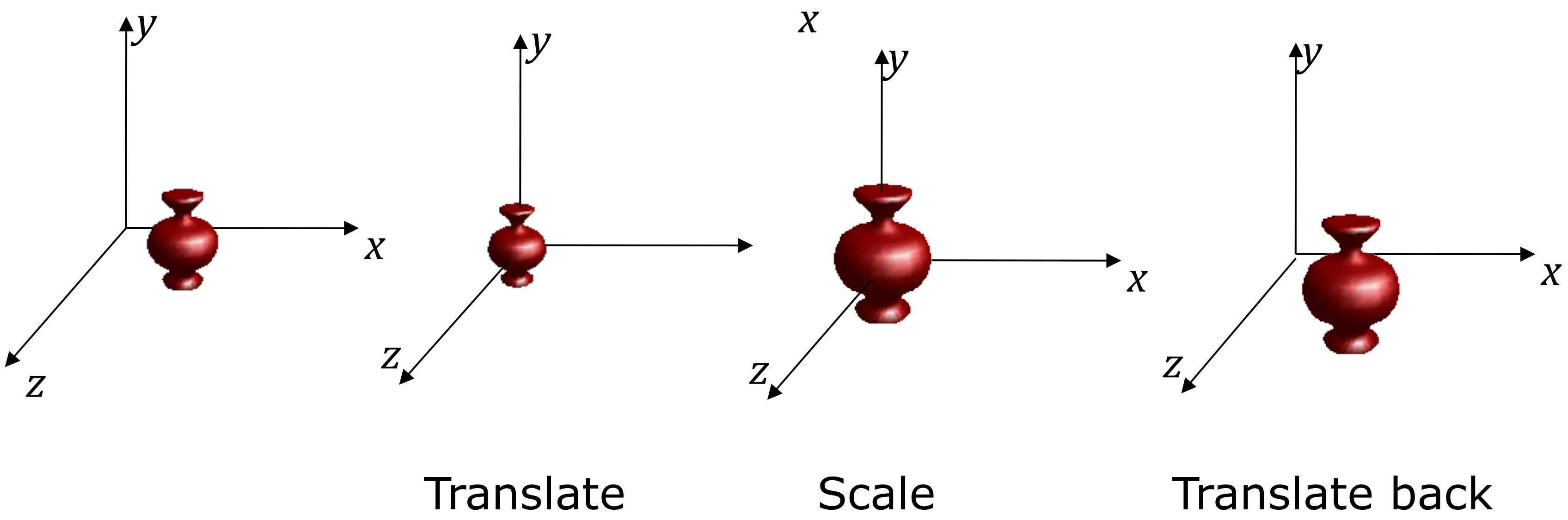




Scaling w.r.t. a Fixed Point

- Translate to origin, scale, translate back

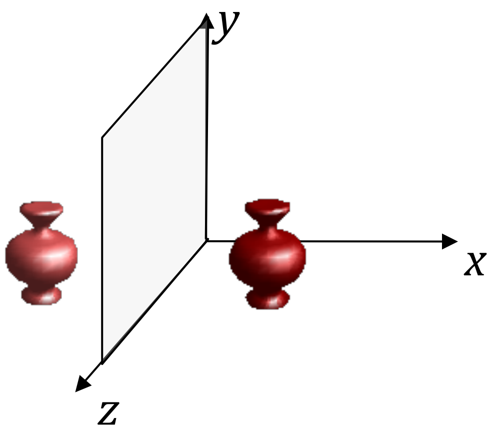
$$\mathbf{P}' = \mathbf{T}(x_f, y_f, z_f) \cdot \mathbf{S} \cdot \mathbf{T}(-x_f, -y_f, -z_f) \cdot \mathbf{P}$$



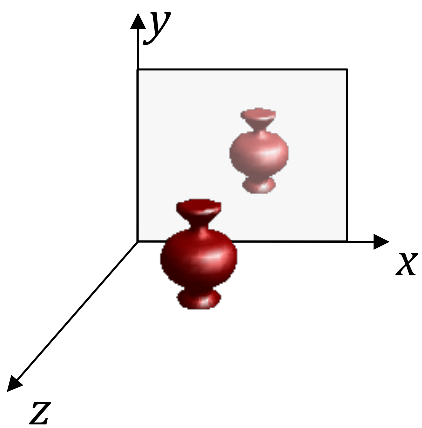
Reflection

- Reflection over the major planes:

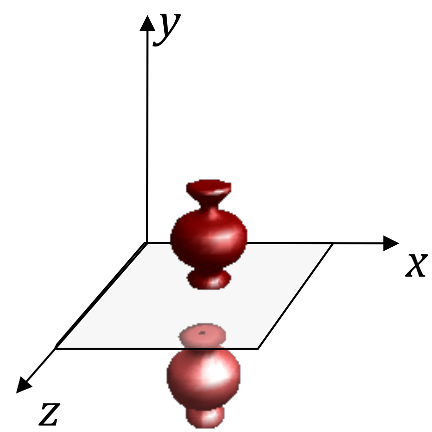
How about reflection over an arbitrary plane?



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



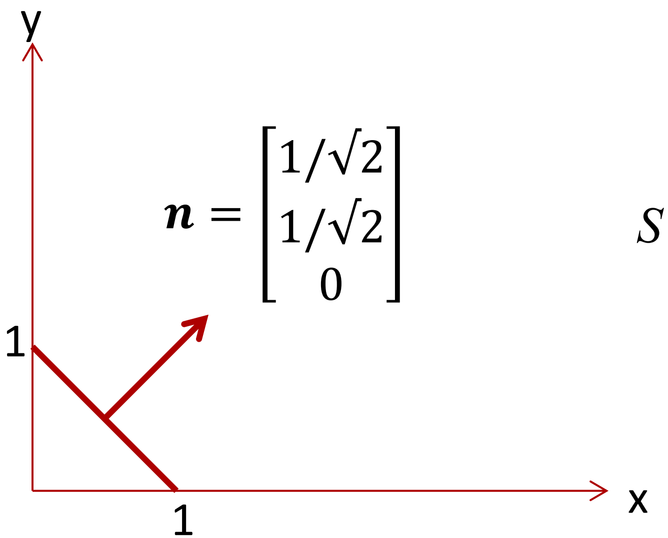
$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



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

Transforming Normals

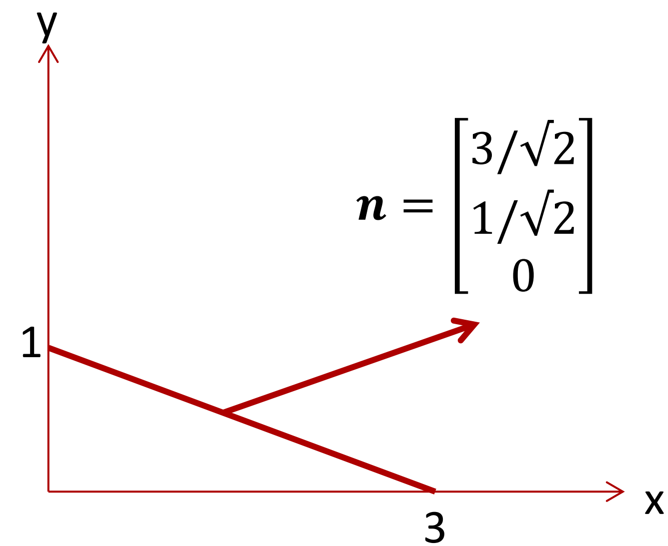
- When we transform an object, what happens to its normals?
- Do they get transformed by the same matrix or does it require a different one?



$$\boldsymbol{n} = \begin{bmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \\ 0 \end{bmatrix}$$

Scale by:

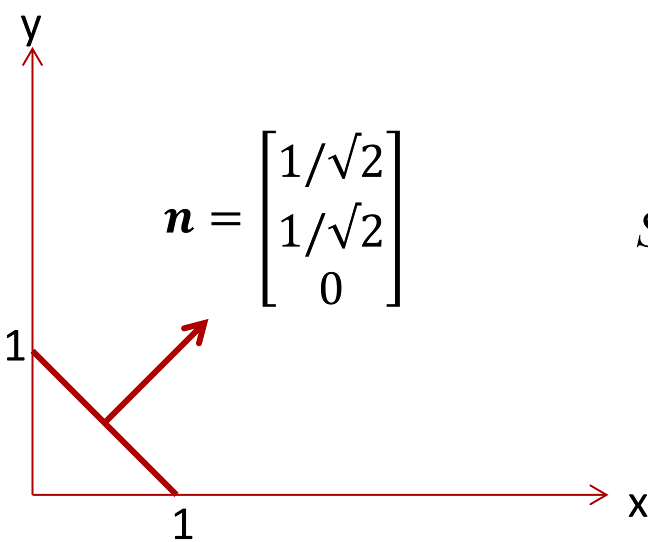
$$S = \begin{bmatrix} 3 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



$$\boldsymbol{n} = \begin{bmatrix} 3/\sqrt{2} \\ 1/\sqrt{2} \\ 0 \end{bmatrix}$$

Transforming Normals

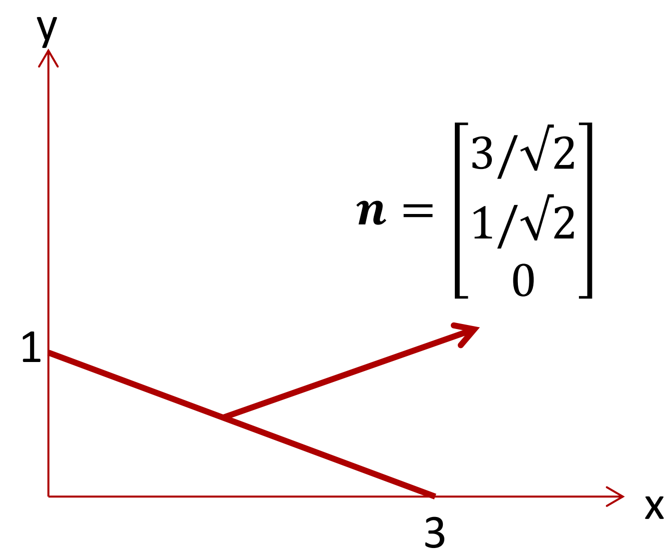
- After the transformation the normal is no longer perpendicular to the object
- Also it is not a unit vector anymore



$$\boldsymbol{n} = \begin{bmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \\ 0 \end{bmatrix}$$

Scale by:

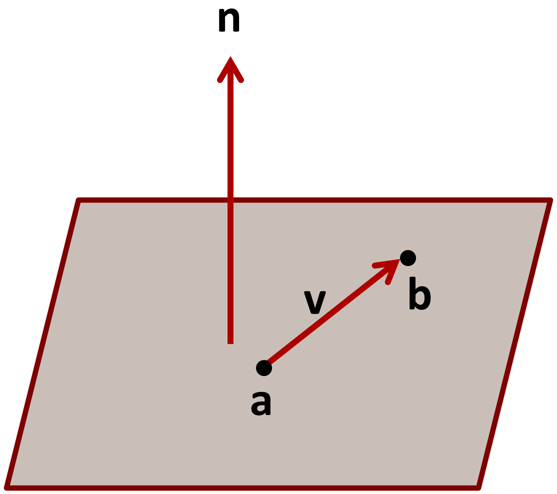
$$S = \begin{bmatrix} 3 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



$$\boldsymbol{n} = \begin{bmatrix} 3/\sqrt{2} \\ 1/\sqrt{2} \\ 0 \end{bmatrix}$$

Transforming Normals

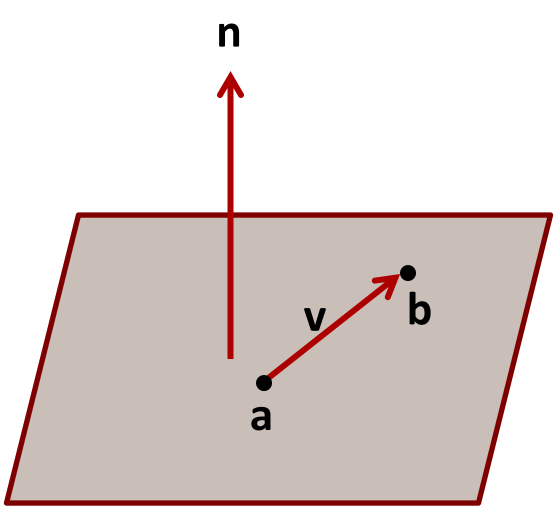
- Rotation and translation has no problems
- But, since all transformations are combined into a single matrix M, we should consider the general case.



- We must have **n.**(**b**-**a**) = **n.v** = 0 and this relationship should be preserved after the transformation

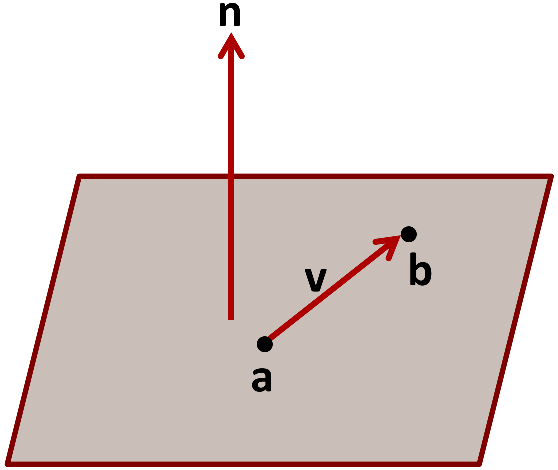
Transforming Normals

- That is **n**.**v** = 0 and **n'**.**v'** = 0 where **v'** = M**v** and **n'** = Z**n**
- Z is the matrix we are looking for
- How to compute Z?



Transforming Normals

- $\mathbf{n}.\mathbf{v} = \mathbf{n}^T\mathbf{v} = 0$
- $\mathbf{n}'.\mathbf{v}' = \mathbf{n}'^T\mathbf{v}' = \mathbf{n}'^T M\mathbf{v} = \mathbf{n}^T Z^T M\mathbf{v} = 0$
- If $Z^TM = I$ (identity) the relationship will be preserved
- So $Z = (M^{-1})^T$
- Note that this is equal to $(M^T)^{-1}$ as $(M^{-1})^T = (M^T)^{-1}$ for a square (n by n) matrix M



A Word on Notation

- Until now, we performed transformations by multiplying our points from the right:

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

- Another notation is to multiply from the left:

$$[x' \quad y' \quad z' \quad 1] = [x \quad y \quad z \quad 1] \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ 0 & 0 & 0 & 1 \end{bmatrix}^T$$

Note that in this case everything is transposed

A Word on Terminology

- Imagine a 2D rotation matrix such as:

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

- Transforming an object by this matrix will not change its shape
- If we also add translation:

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

- The shape will remain intact

A Word on Terminology

- In general, an arbitrary sequence of rotation and translation matrices will have the following form:

$$\begin{bmatrix} r_{11} & r_{12} & t_x \\ r_{21} & r_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

- Such transformations are called rigid-body transformations
- A shape may be rotated and translated by its form is not altered in any way

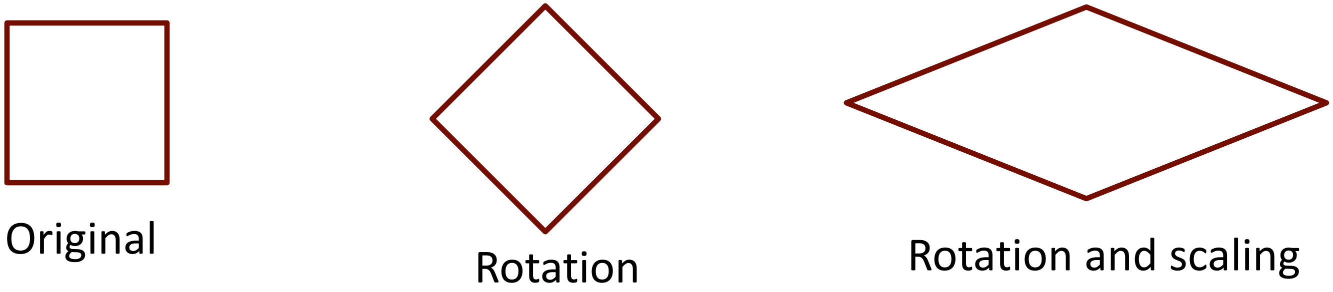
A Word on Terminology

- Imagine also adding scaling
- The matrix will now look like:

$$\begin{bmatrix} a & b & t_x \\ c & d & t_y \\ 0 & 0 & 1 \end{bmatrix}$$

where a, b, c, d contain the effect of rotation and scaling combined

- Such transformations will not necessarily preserve lengths and angles, but parallel lines will remain parallel



A Word on Terminology

- Such transformations are called affine transformations
- An arbitrary sequence of rotation, translation, scaling, and shearing will produce an affine transformation
- Note that we still have some degrees of freedom left in the last row of our matrix:

$$\begin{bmatrix} a & b & t_x \\ c & d & t_y \\ \boxed{0 \quad 0} & & 1 \end{bmatrix}$$

- By using this we can create projective transformations in which parallel lines may no longer be parallel