

# Computer Graphics

Basic Mathematics: Linear Transformations

**Andreas Aristidou**

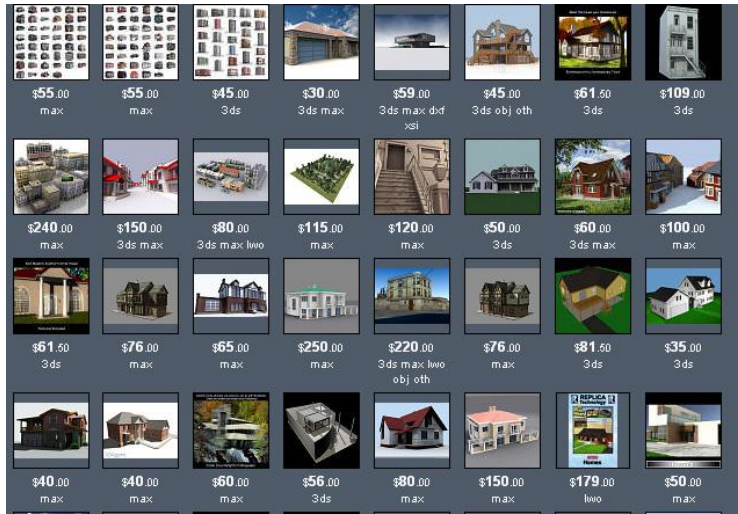
andarist@ucy.ac.cy

<http://www.andreasaristidou.com>

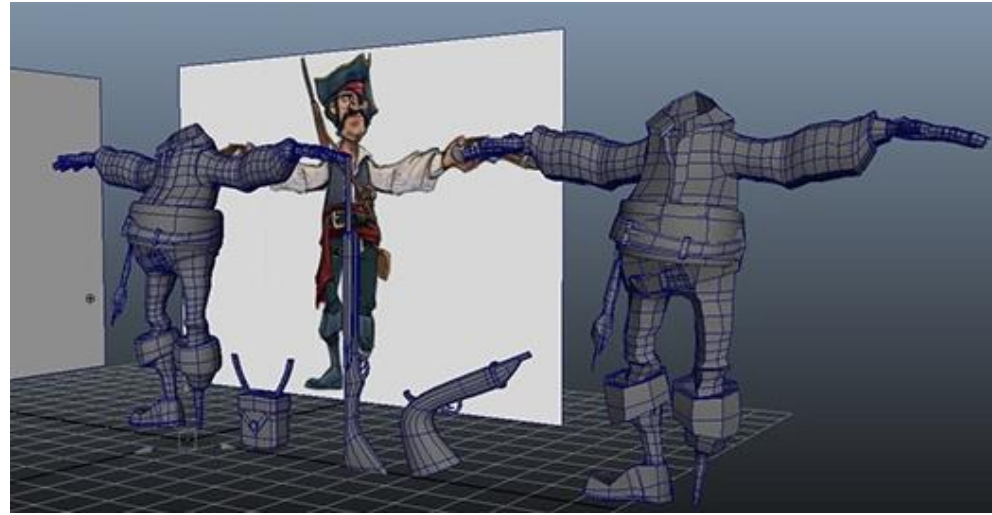
## A wireframe model of a cow, showing the underlying mesh structure of the animal's body, head, and legs. The model is composed of numerous interconnected lines forming a complex geometric shape that represents the cow's form. The cow is standing on all fours, facing right, with its head slightly lowered. The wireframe is dense, particularly in the areas of the head, neck, and legs, indicating a high level of detail in the mesh. The overall structure is symmetrical, with the left and right sides of the cow's body mirroring each other. The wireframe is rendered in a dark gray or black color against a plain white background.



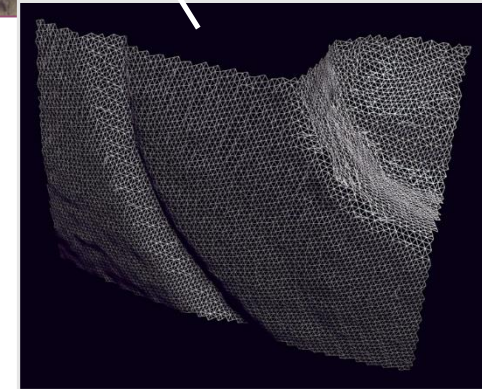
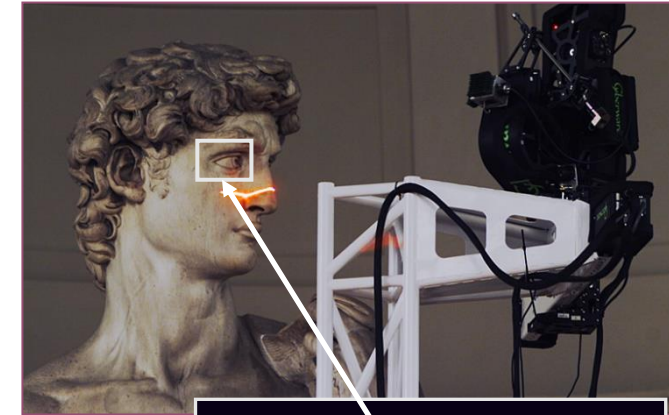
# Geometric Modeling: Introduction



3D Model Library



3D Software



3D Laser Scan

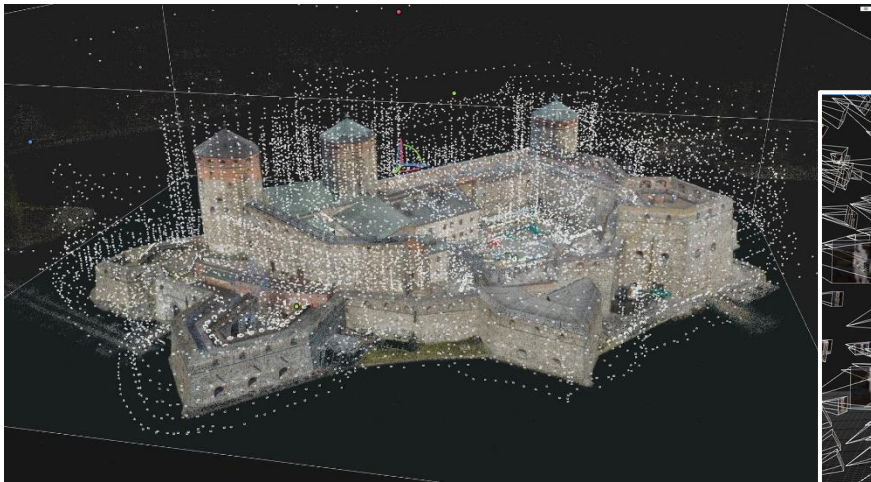
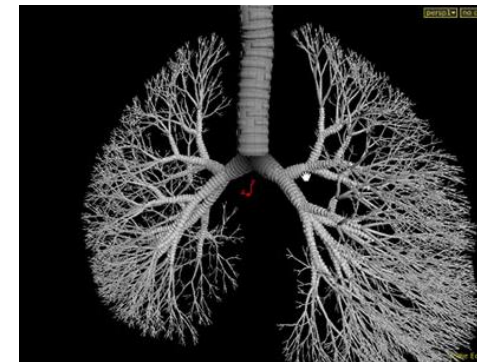
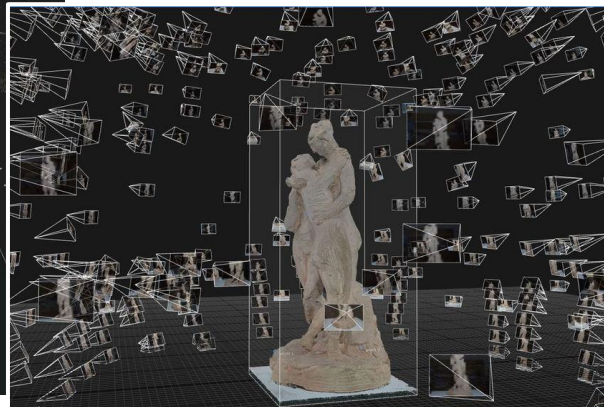


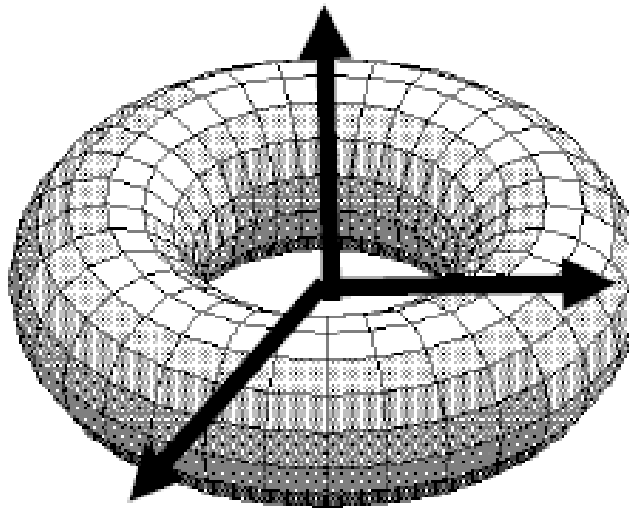
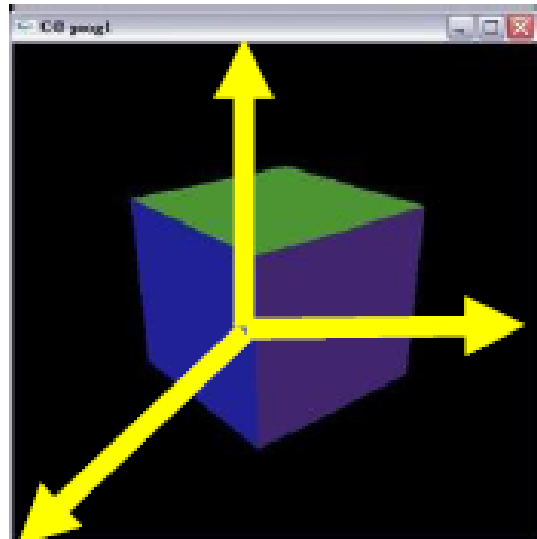
Image-based: *Photogrammetry*



Procedural Modeling

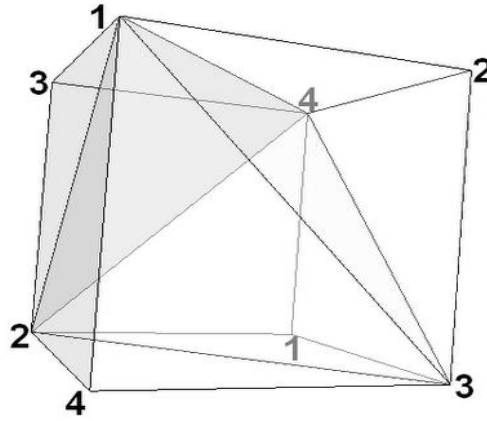
# Set up objects in the scene

- Once the modeling of the objects/characters is complete, they must be placed in the environment
- Therefore, we need to know the vertex positions of the objects (reference point) in the world coordinate system.
- Usually objects are defined in their own local coordinate system.



# How do we use geometric transformations?

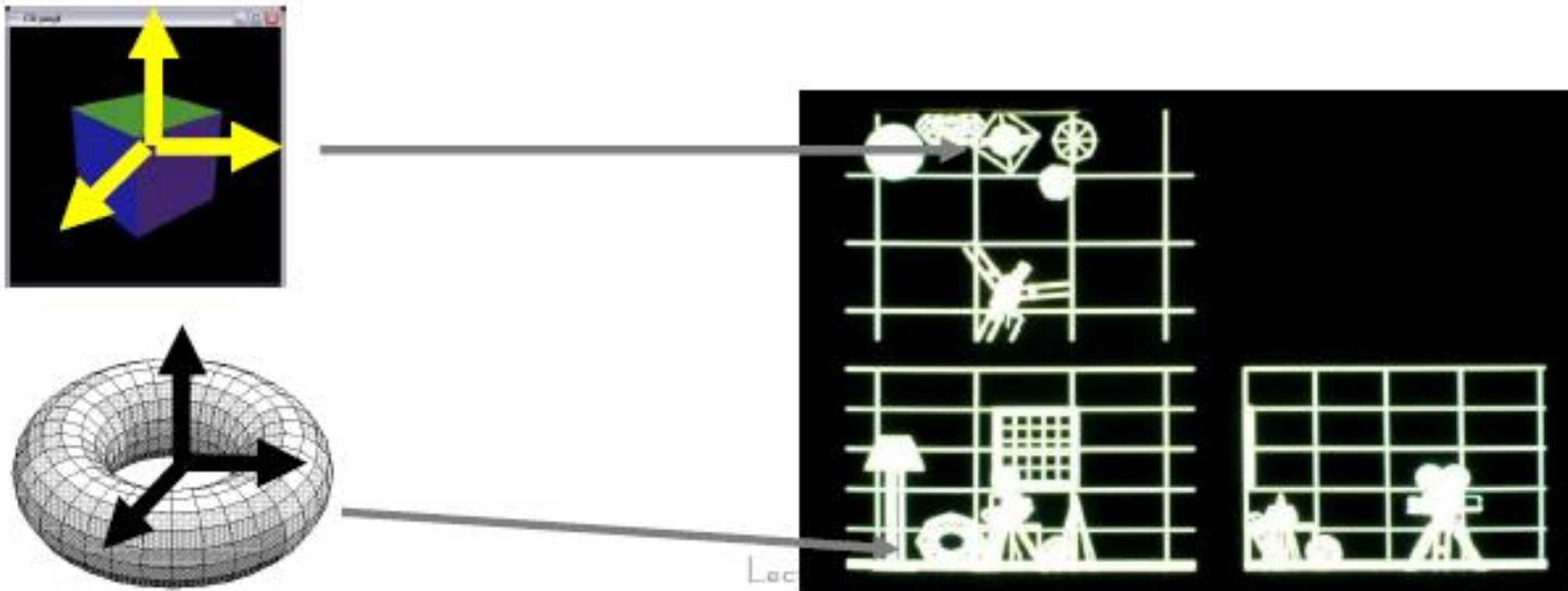
- Objects in a scene at the lowest level are a collection of vertices...



- These objects have a location, orientation, size
- Correspond to transformations: Translation ( $\mathbf{T}$ ), Rotation ( $\mathbf{R}$ ), and Scaling ( $\mathbf{S}$ )

# Geometric transformations

- So, our goal is to transfer, rotate and/or scale the vertices in the global coordinate system.





# Transformations

- The most important transformations are
  - ***translation*** (move the object in a straight line),
  - ***scaling*** (change its dimensions),
  - ***rotation*** (rotate based on a point),
  - ***reflection*** (reflection based on a straight line),
  - ***shearing*** (warp the shape of an object based on its axes).

# Transformations



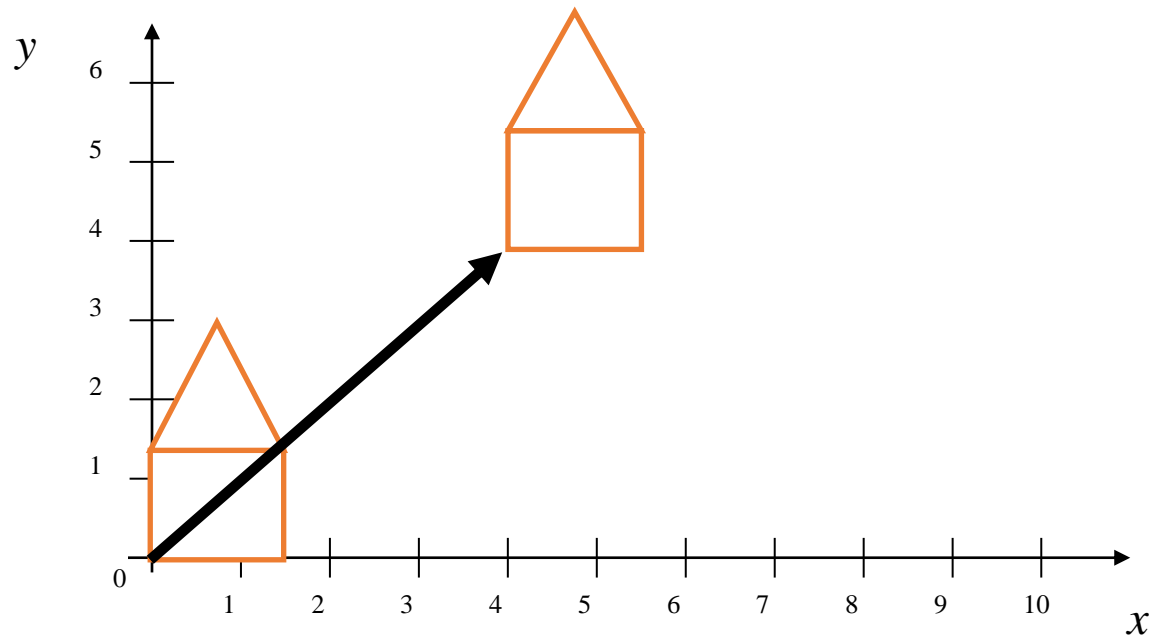
An object can be scaled, rotated, reflected (flipped), warped, and translated.



# 2D Translation

- Simply move an object from one location to another

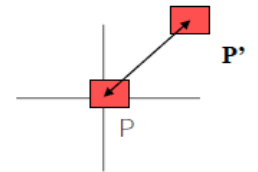
- $x' = x + dx$        $y' = y + dy$



**Note:** The house moves in relation to the origin of the axes

It can also be expressed as matrices!

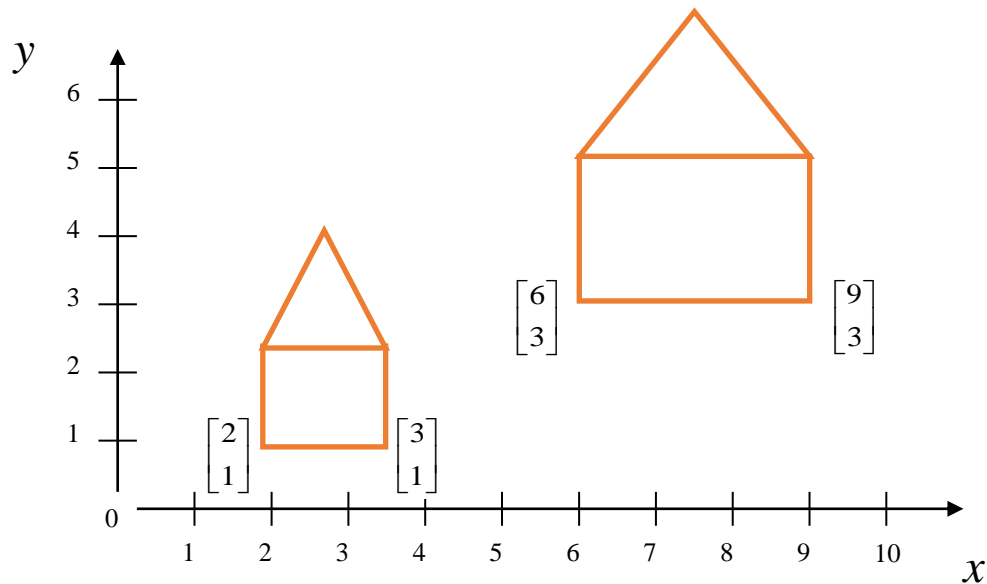
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} d_x \\ d_y \end{bmatrix}$$



# 2D Scaling

- In scalar scaling we multiply all the coordinates
- **Attention!:** The objects get bigger and change position!

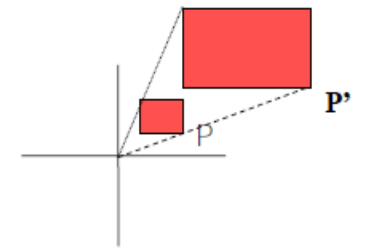
$$x' = S_x \times x \quad y' = S_y \times y$$



Same in matrices!

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x x \\ s_y y \end{bmatrix}$$



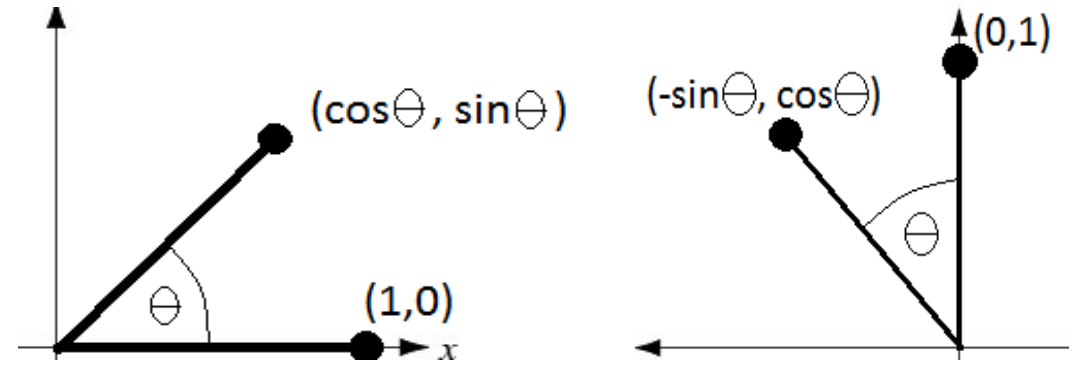
**Note:** The house moves in relation to the origin of the axes

# Rotation

- Rotates all coordinates at a specified angle

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

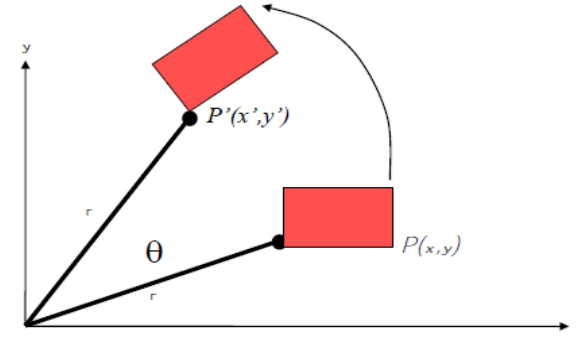
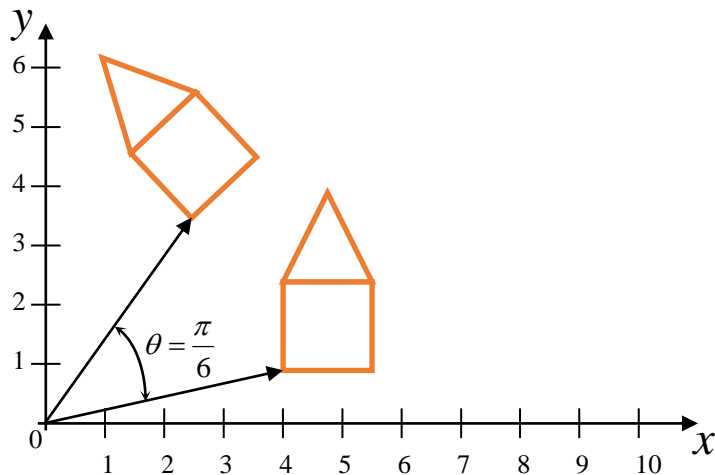
- Points **always** rotate around the origin of the axes



Same in matrices!

Rotation

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



# Basic Transformations

- Translation

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} d_x \\ d_y \end{bmatrix}$$

- Scale

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

- Rotation

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

- Different equations for each transformation
- Homogeneous Transformations!



# Homogeneous Transformations

- We add an additional dimension, which is equal to 1 ( $x, y, z, 1$ )
- Why?
  - Using homogeneous transformation in 2D (or in 3D), transformations can be described by multiplying with a  $3 \times 3$  matrix (or  $4 \times 4$ ).

# Homogenous Transformations

- Translation

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & d_x \\ 0 & 1 & d_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x + d_x \\ y + d_y \\ 1 \end{bmatrix}$$

- Scale

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

- Rotation

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

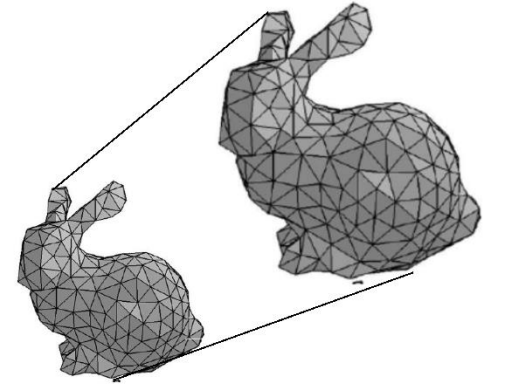
# Inverse Transformations

- Transformations can be easily reversed using the inverse transformation.
  - When we want to recall a transformation
  - It is possible due to the homogeneous expressions

$$T^{-1} = \begin{bmatrix} 1 & 0 & -dx \\ 0 & 1 & -dy \\ 0 & 0 & 1 \end{bmatrix}$$
$$S^{-1} = \begin{bmatrix} \frac{1}{s_x} & 0 & 0 \\ 0 & \frac{1}{s_y} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$
$$R^{-1} = \begin{bmatrix} \cos \theta & \sin \theta & 0 \\ -\sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

# Apply transformations

- **Attention!** The multiplication of matrices is NOT commutative!



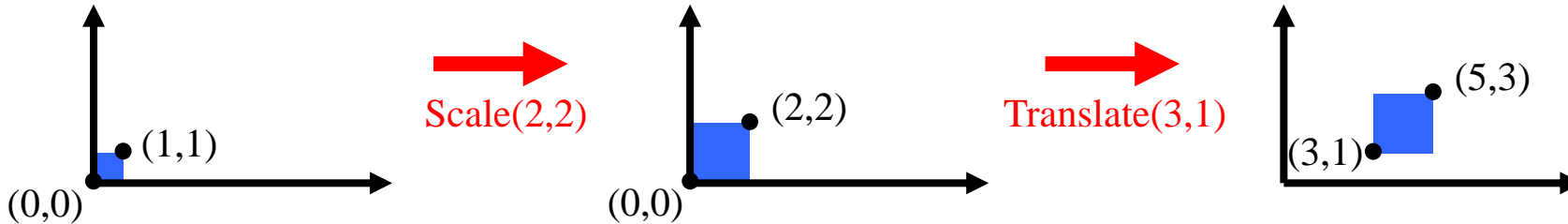
$$AB \neq BA$$

- When applying such multiplication of matrices, one important thing to remember is that they are not commutative; the change in the order of multiplication will result in different coordinates.
- For example, consider the example of translating and scaling an object:



# Apply transformations

- Scale and translate



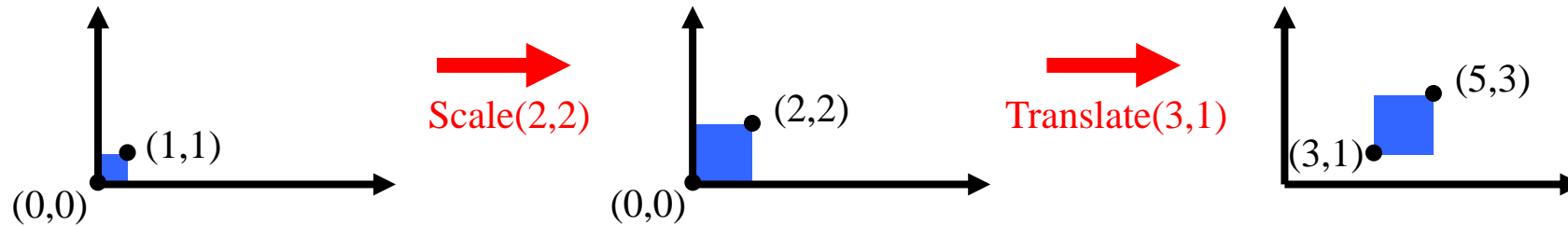
- Use matrix multiplication:  $p' = T(S p) = (TS)p$

$$TS = \begin{bmatrix} 1 & 0 & 3 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 0 & 3 \\ 0 & 2 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

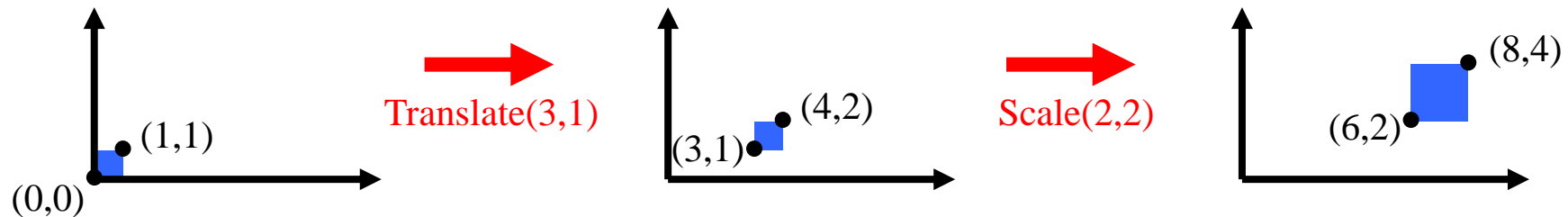
You can see that the effect of scaling before translating, is different from when the translation is done first, and then the scaling.

# Apply transformations

Scale and then Translate:  $p' = T(S p) = (TS)p$



Translate and then Scale:  $p' = S(T p) = (ST)p$



You can see that the effect of scaling before translating, is different from when the translation is done first, and then the scaling.

# Apply transformations

Scale and then Translate:  $p' = T(S p) = (TS)p$

$$TS = \begin{bmatrix} 1 & 0 & 3 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 0 & 3 \\ 0 & 2 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$

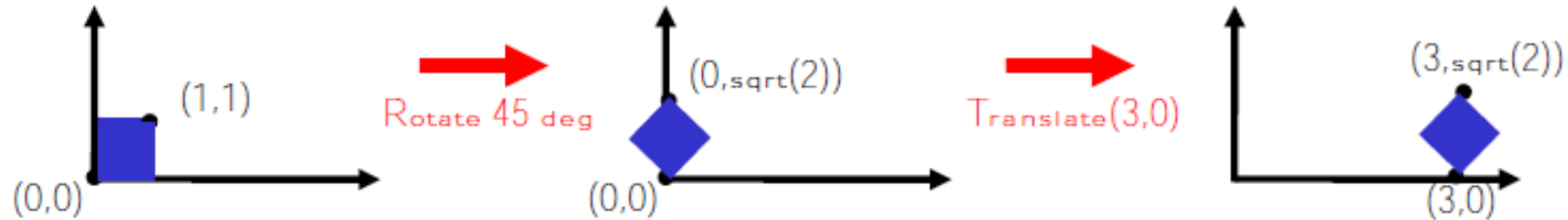
Translate and then Scaling:  $p' = S(T p) = (ST)p$

$$ST = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 3 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 0 & 6 \\ 0 & 2 & 2 \\ 0 & 0 & 1 \end{bmatrix}$$

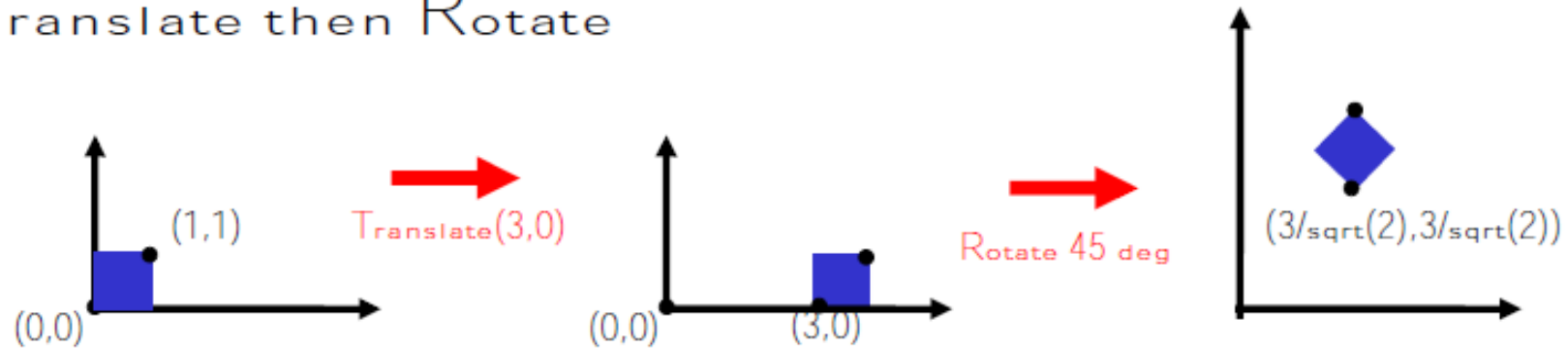
If we look at the transformation matrices, you can see that the elements are different when the order of multiplication is different.

# Apply transformations

Rotate then Translate



Translate then Rotate



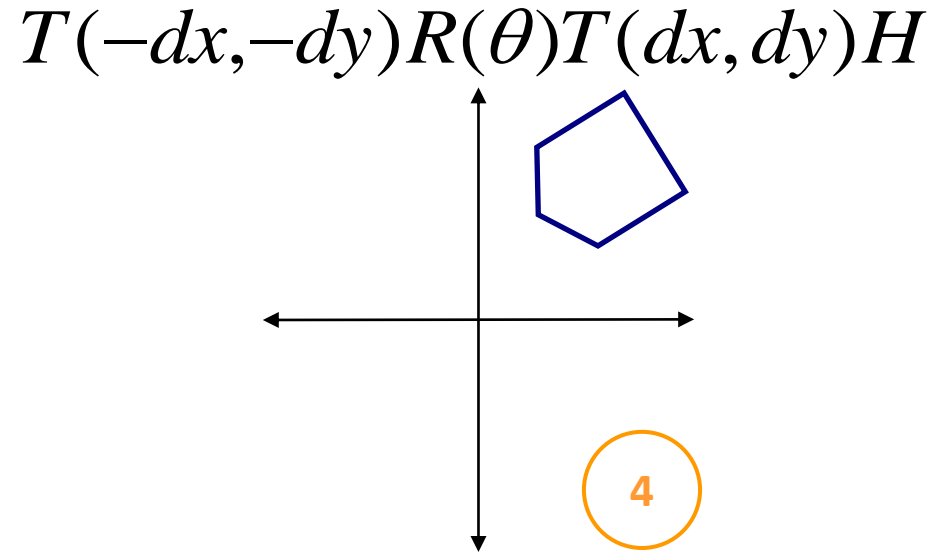
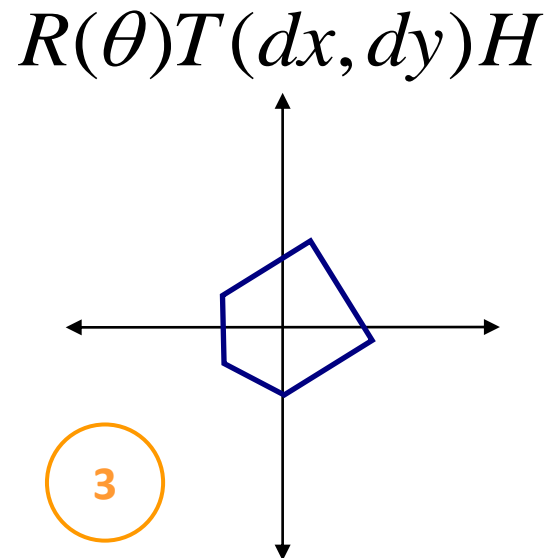
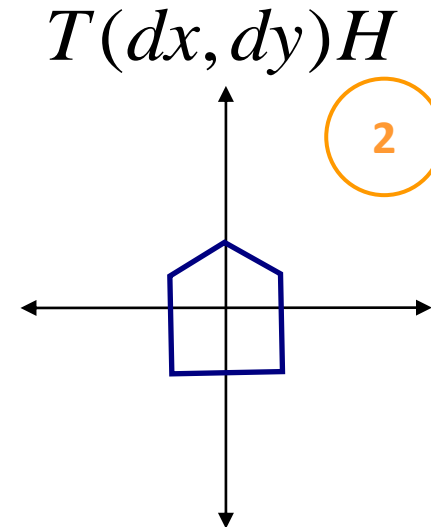
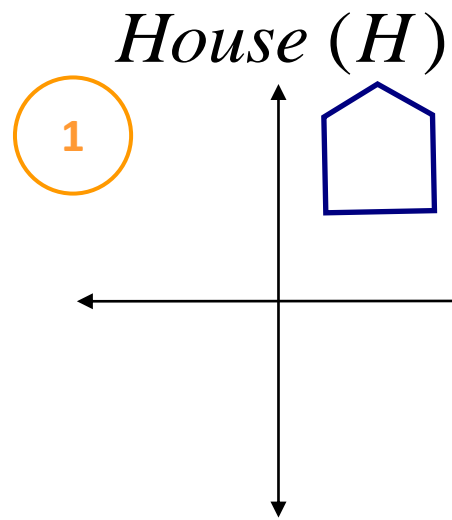
Here is another example where first the rotation is done and then the translation. As before, the end result is different if first the translation is done and then the rotation.



# Apply transformations

- A number of transformations can be combined into one matrix to make things easier
- It is allowed by the fact that we use homogeneous coordinates
- Suppose you rotate a polygon around a point (not the origin of the axes)
- Transfer the point to the origin of the axes
  - Rotate around the point
  - Translating the point back

# Apply transformations



# Apply transformations

- The three transformations are expressed in matrices as follows

$$\begin{bmatrix} 1 & 0 & -dx \\ 0 & 1 & -dy \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & dx \\ 0 & 1 & dy \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

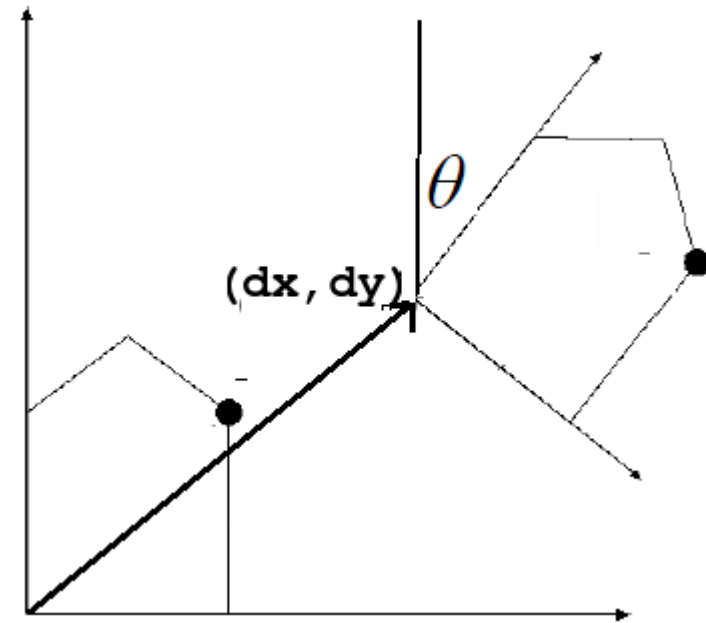
$$v' = T(-dx, -dy)R(\theta)T(dx, dy)v$$

**WARNING: the multiplication of the matrices is not commutative, so the order matters**

# Rotation, Translation

- The matrix of rotation and then translation is often used so it may be worth to remember it.
- It is quite simple ; just add the translation elements in the 3rd column of the rotation matrix and thats it.
- So if you want to put an object at position  $(d_x, d_y)$  and rotate it around its local origin for  $\theta$  degrees, you need to use this matrix.

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

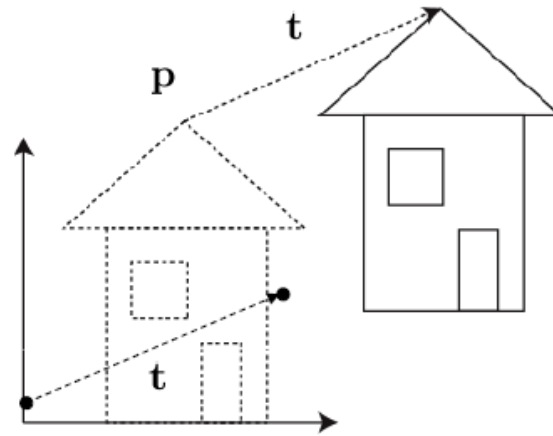




# 3D Translation

- Homogeneous transformation matrices in 3D

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



$$\mathbf{p} = \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix} \quad \mathbf{t} = \begin{bmatrix} t_x \\ t_y \\ t_z \\ 0 \end{bmatrix}$$

$$\mathbf{p}' = \mathbf{p} + \mathbf{t} = \begin{bmatrix} p_x + t_x \\ p_y + t_y \\ p_z + t_z \\ 1 \end{bmatrix}$$

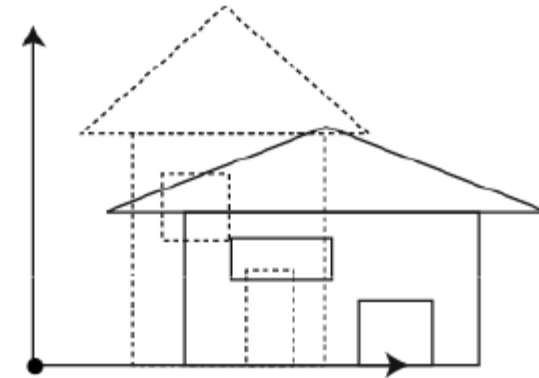
$$\begin{bmatrix} p'_x \\ p'_y \\ p'_z \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix}$$

$$\mathbf{p}' = \mathbf{T}(\mathbf{t})\mathbf{p}$$

# 3D Scaling

- Similarly, the matrix for scalar scaling is:

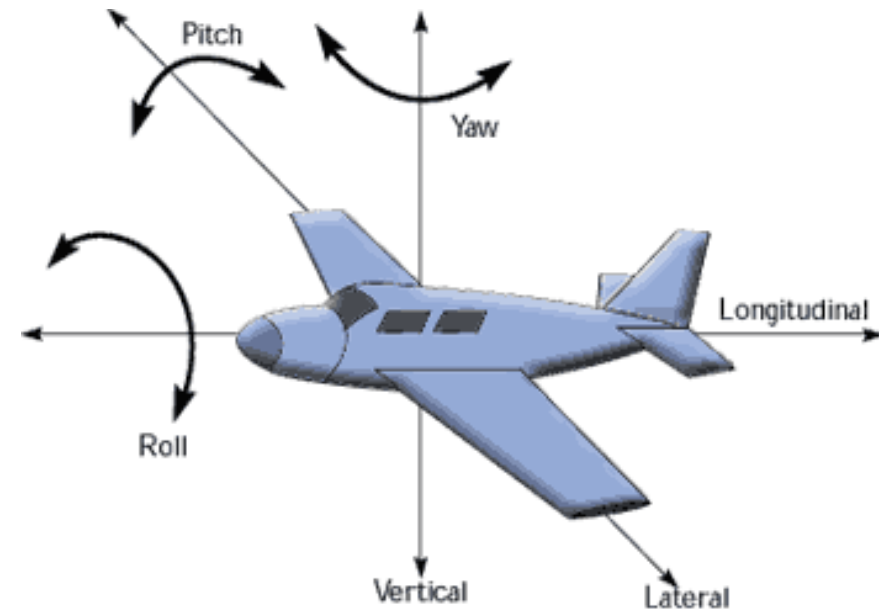
$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$



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

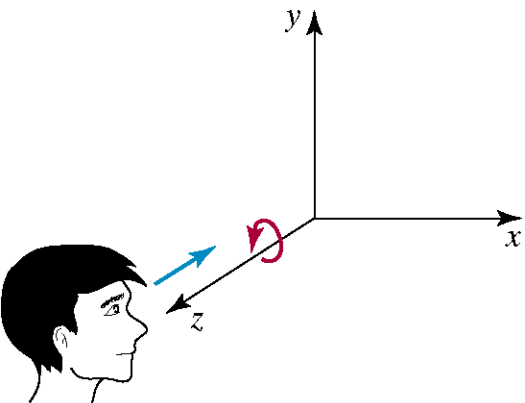
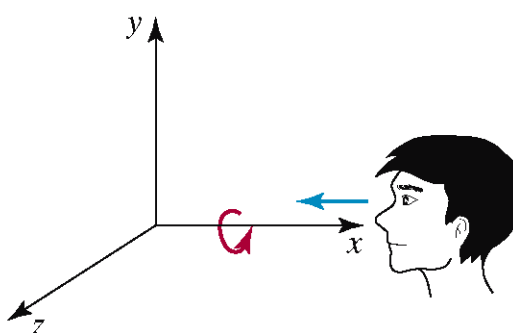
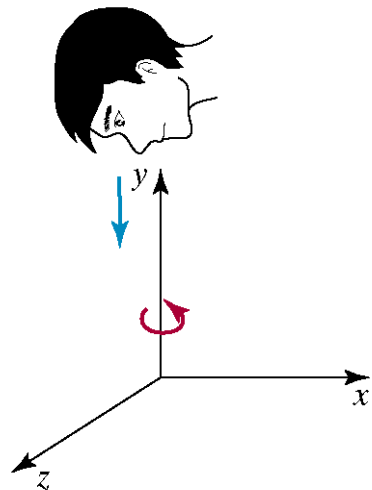
# 3D Rotation

- When we performed rotations in two dimensions, we only had the option of rotation for the z-axis.
- In the case of the three dimensions we have more options:
  - Rotate on the axis of  $x$  – **pitch**
  - Rotate on the axis of  $y$  – **yaw**
  - Rotate on the axis of  $z$  – **roll**



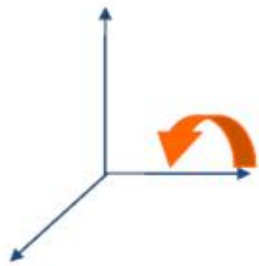
# 3D Rotation

- The equations for the three types of rotations in 3D are as follows:


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

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

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

# 3D Rotation

- Rotate around the  $x$  axis -  $R_x(\theta)$

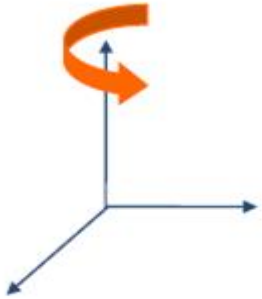


$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta & 0 \\ 0 & \sin \theta & \cos \theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

Note that the  $x$  values do not change as the  $y$  and  $z$  values change.

# 3D Rotation

- Rotate around the  $y$  axis -  $R_y(\theta)$

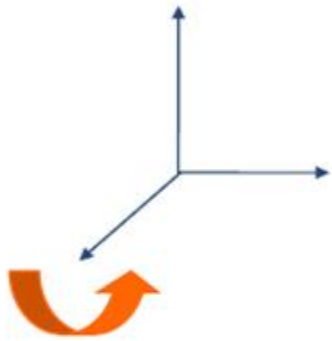


$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} \cos \theta & 0 & \sin \theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin \theta & 0 & \cos \theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$



# 3D Rotation

- Rotate around the  $z$  axis -  $R_z(\theta)$



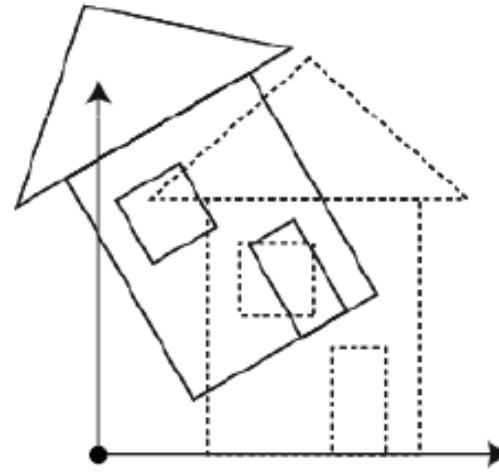
$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} \cos \theta & -\sin \theta & 0 & 0 \\ \sin \theta & \cos \theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

# 3D Rotation

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

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

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



# 3D Rotation

- The total rotation is

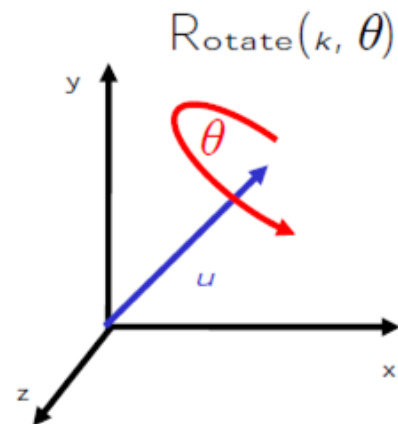
$$\mathbf{R}_{x,y,z}(\theta_x, \theta_y, \theta_z) = \mathbf{R}_x(\theta_x)\mathbf{R}_y(\theta_y)\mathbf{R}_z(\theta_z)$$

- The angles  $\theta_x, \theta_y, \theta_z$  are called Euler angles

$$\mathbf{R}_x(\theta_x)\mathbf{R}_y(\theta_y)\mathbf{R}_z(\theta_z) \neq \mathbf{R}_z(\theta_z)\mathbf{R}_y(\theta_y)\mathbf{R}_x(\theta_x)$$

# Rotate on a random (arbitrary) axis

- About  $(u_x, u_y, u_z)$ , a unit vector on an arbitrary axis



$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} u_x u_x (1-c) + c & u_z u_x (1-c) - u_z s & u_x u_z (1-c) + u_y s & 0 \\ u_y u_x (1-c) + u_z s & u_z u_x (1-c) + c & u_y u_z (1-c) - u_x s & 0 \\ u_z u_x (1-c) - u_y s & u_y u_z (1-c) + u_x s & u_z u_z (1-c) + c & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

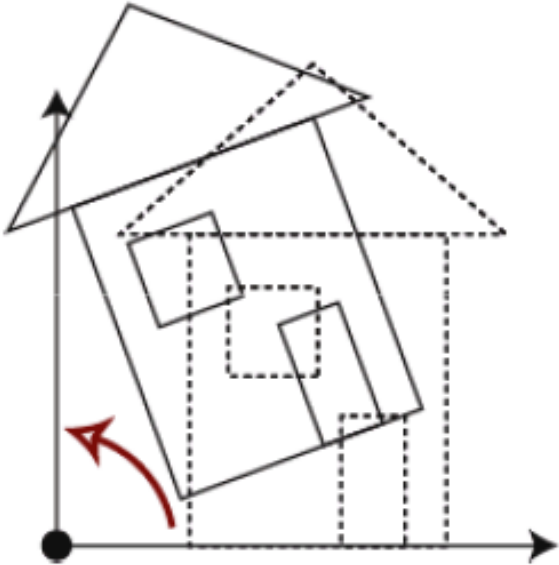
where  $c = \cos \theta$  &  $s = \sin \theta$

# Inverse Rotation

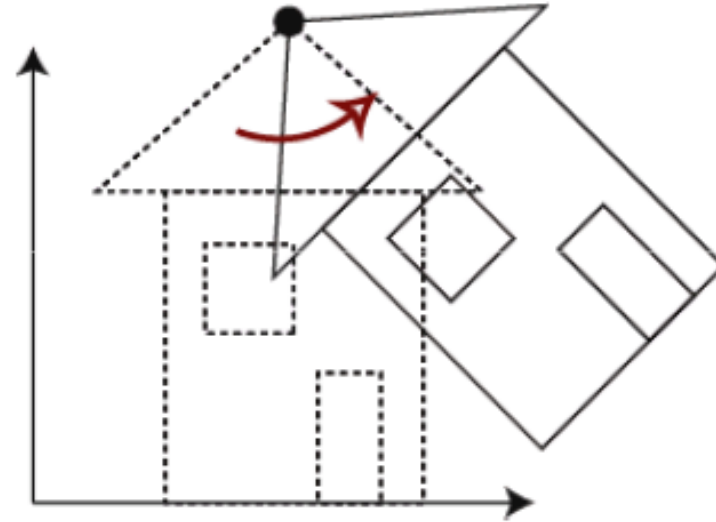
- The inverse rotation can be calculated by the **transpose** matrix.

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

# Rotate around a point

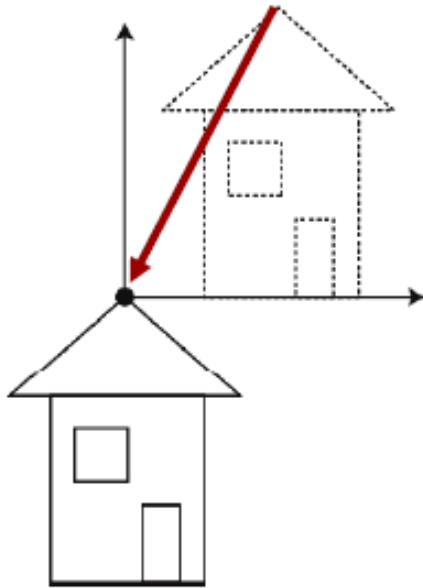


Rotation around  
origin

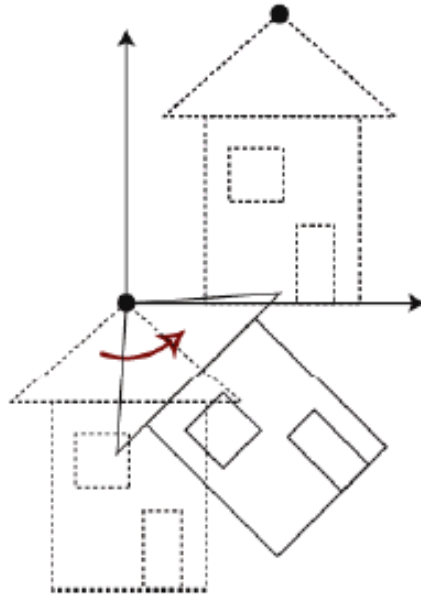


Rotation with  
pivot

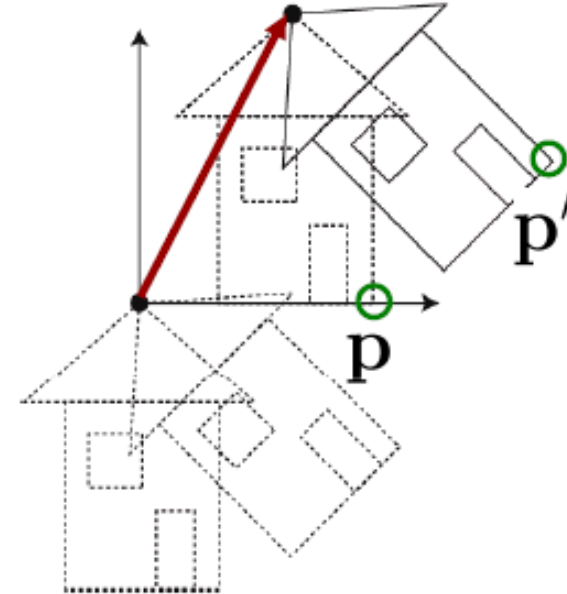
# Rotate around a point



1. Translation  $T$



2. Rotation  $R$



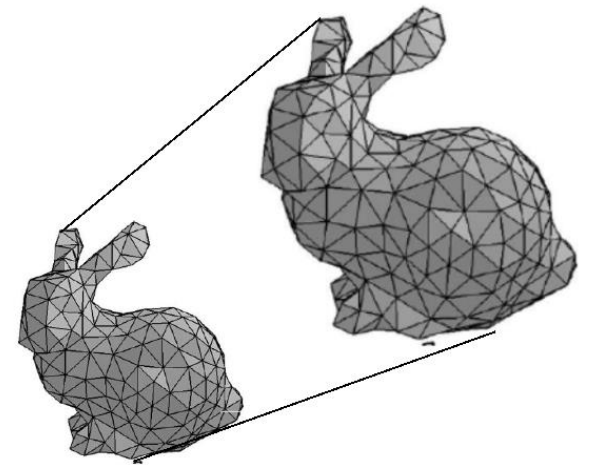
3. Translation  $T^{-1}$

$$p' = T^{-1}RTp$$



# Applying multiple transformations together

- We can apply the aforementioned transformations to move, scale, and rotate the vertices of objects.
- What if we want a combination of many such transformations?



# Adding all this together

- $R$  are the elements of rotation and scaling,
- $T$  are the elements of the translation.

$$\begin{pmatrix} R_1 & R_2 & R_3 & T_1 \\ R_4 & R_5 & R_6 & T_2 \\ R_7 & R_8 & R_9 & T_3 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

## 3D Rotation: Problems (issues) – Gimbal Lock

- Problem occurs when two of the rotation axes are aligned. This is called «**Gimbal Lock**».
- Locking the axes is a key problem when representing 3D rotation with Euler angles or fixed angles.
- Phenomenon in which two axes of rotation of an object are aligned.
  - A degree of freedom is lost
  - Simply put, it means that the object will not rotate as you think it will rotate.
- **Solution:** Use of another system (e.g., quaternions)

