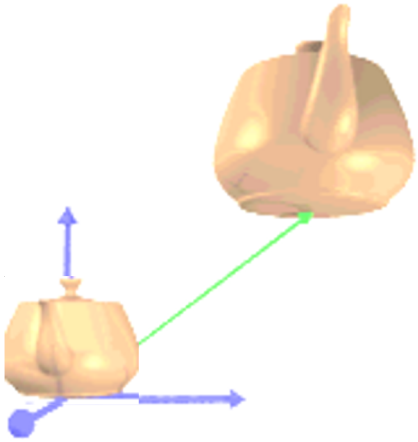
FACULTY OF ENGINEERING AND ARCHITECTURE

E016712: Computer Graphics

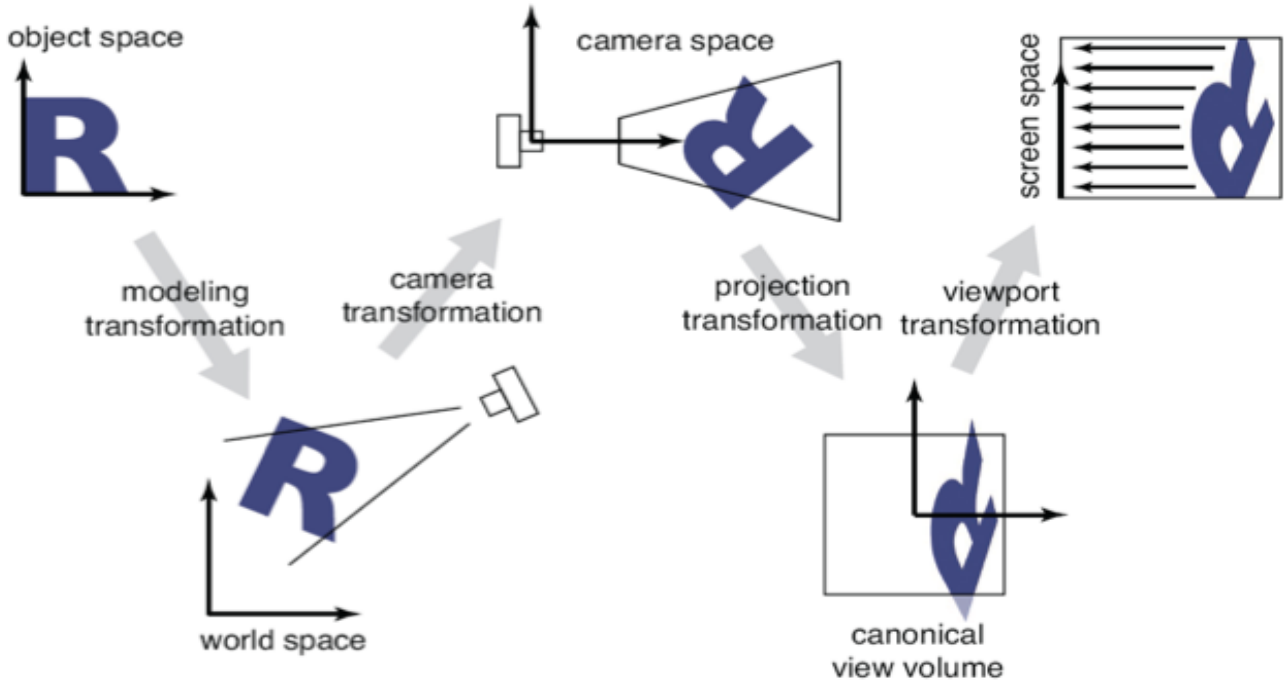
Transformations



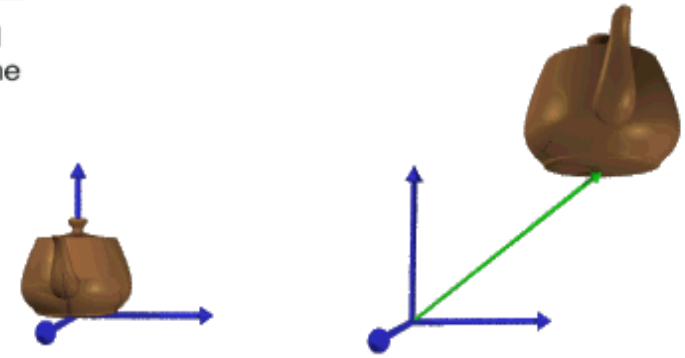
Lecturers: Aleksandra Pizurica and Danilo Babin

GHENT UNIVERSITY

Transformations in computer graphics



- Goal: introduce methodology to
  - Change coordinate system
  - Move and deform objects



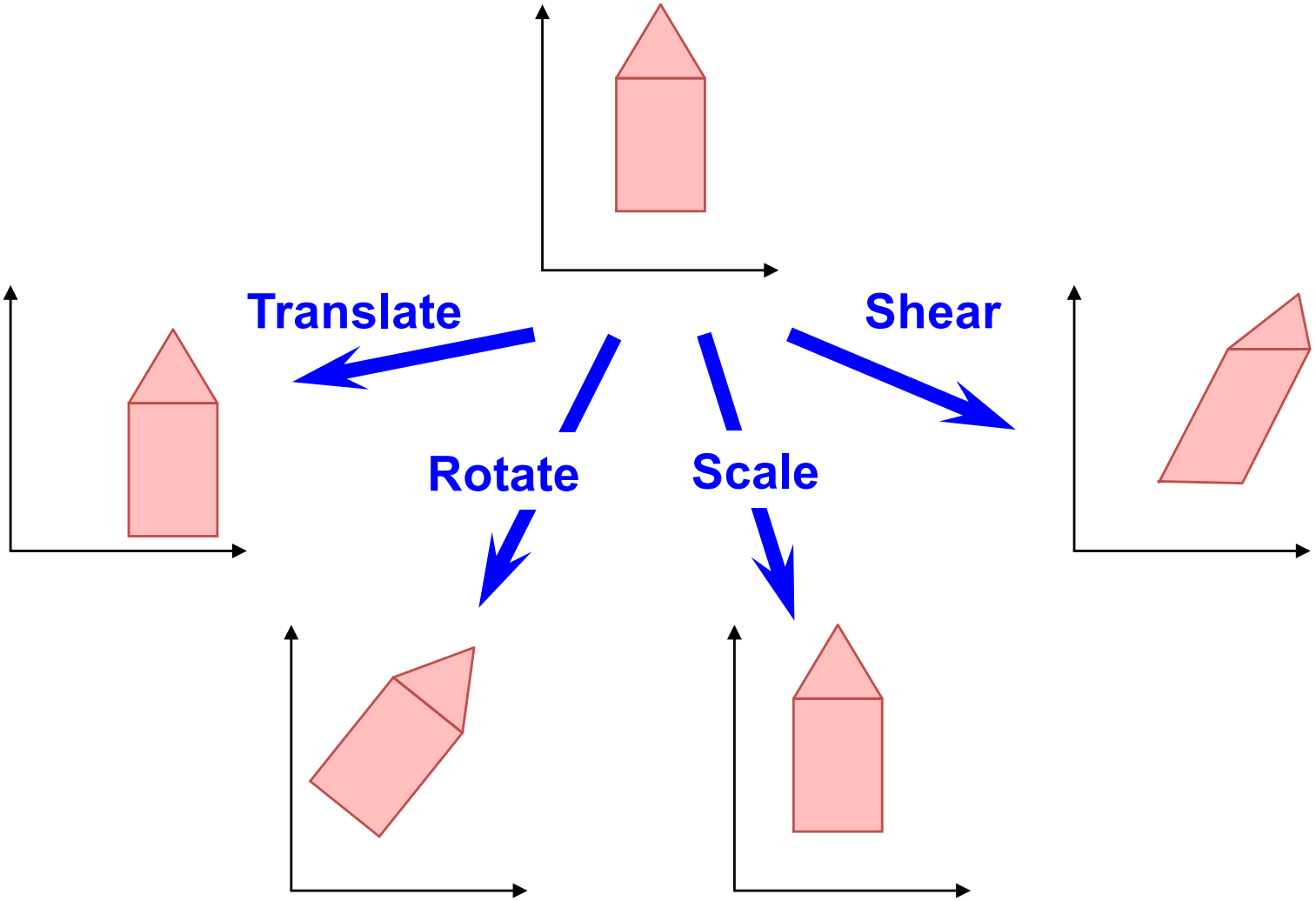
- Principle: transformations are applied to object vertices
  - In 2D, point P(X,Y) is transformed to P'(X',Y'); in 3D, P(X,Y,Z) → P'(X',Y',Z')

Overview

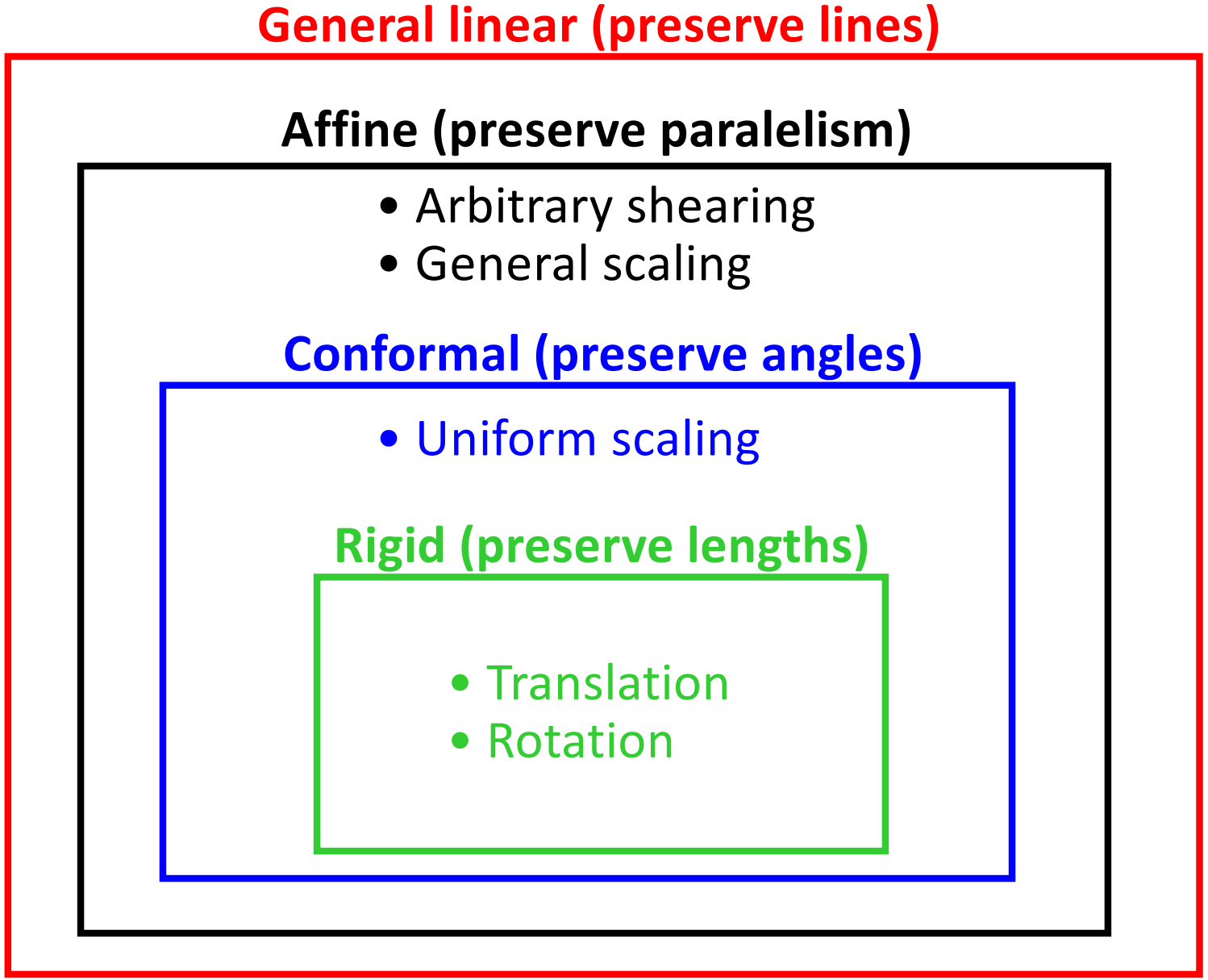
- 2D transformations
- 3D transformations
- Quaternions
- Transformations in OpenGL

2D transformations

2D affine transformations



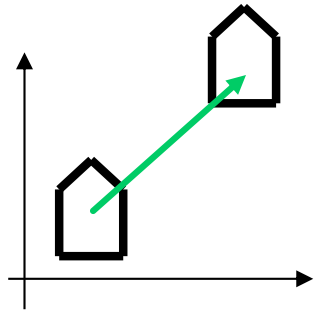
Basic classes of geometric transformations



Elementary 2D transformations

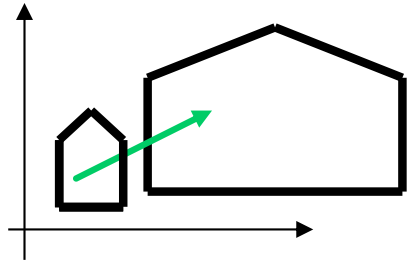
Translation $\mathcal{T}(T_x, T_y)$

$$X' = X + T_X$$
$$Y' = Y + T_y$$



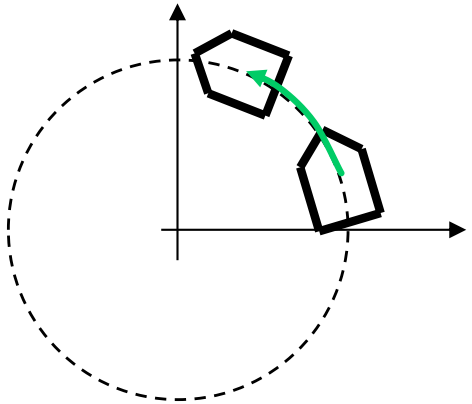
Scaling $\mathcal{S}(S_x, S_y)$

$$X' = X * S_X$$
$$Y' = Y * S_Y$$



Rotation $\mathcal{R}(\theta)$

$$X' = X * \cos\theta - Y*\sin\theta$$
$$Y' = X * \sin\theta + Y*\cos\theta$$



Matrix representation

- Suppose we represent a 2D transformation by a matrix

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

transformed point

transformation matrix

original point

$$x' = ax + by$$
$$y' = cx + dy$$

- Why is this useful?

  -A sequence of transformations → matrix multiplication

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

- Can we do it for any affine transform?

2x2 Matrices

- Which transformations can be represented by 2x2 matrices?

  - Let's look at some examples:

2D identity

- x' = x
- y' = y

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

2D mirror over Y axis

- x' = -x
- y' = y

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

2D scale around (0,0)

- $x' = S_x x$
- $y' = S_y y$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

2D mirror over (0,0)

- x' = -x
- y' = -y

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

2x2 Matrices

- Which transformations can be represented by 2x2 matrices?

  - Let's look at some examples:

2D rotation around (0,0)

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

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

2D Shear

- $x' = x + H_X y$
- $y' = y + H_Y x$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & H_x \\ H_y & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

2x2 Matrices

- Which transformations can be represented by 2x2 matrices?

  - Can we represent translation by a 2x2 matrix?

2D Translation

- $x' = x + T_x$
- $y' = y + T_y$

NO 2x2 matrix!

- Only linear 2D transformations can be represented by a 2x2 matrix

- Linear transformations
  - satisfy: $\mathcal{T}(s_1P_1 + s_2P_2) = s_1\,\mathcal{T}(P_1) + s_2\,\mathcal{T}(P_2)$
  - are combinations of scale, rotation, shear and mirror

Matrix representation for affine 2D transforms

- We want a representation where 2D translation is also represented by a matrix (so that we can easily combine different transformations by multiplication)
- 2D matrix representation of translation does not exist!
- What do we do?
- Solution: use a 3x3 matrix

2D Translation

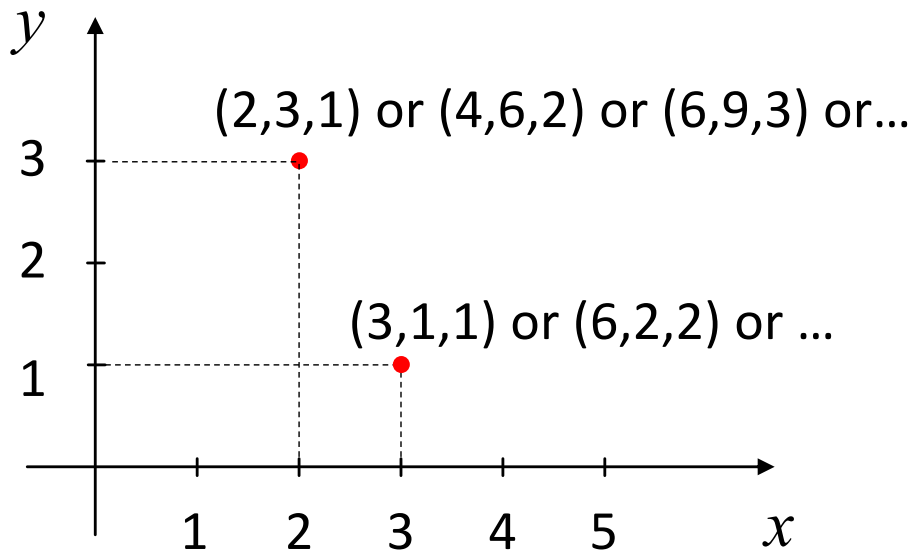
- $x' = x + T_x$
- $y' = y + T_y$

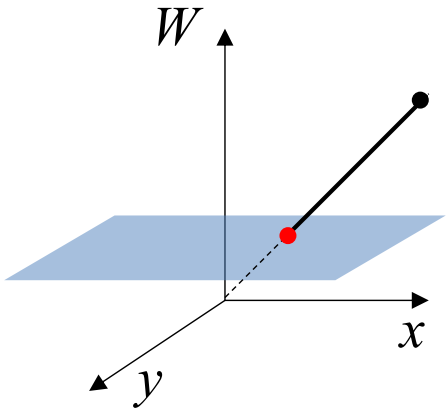
$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & T_x \\ 0 & 1 & T_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Homogeneous coordinates

Homogeneous coordinates

- An equivalent formulation: adding a 3rd coordinate to 2D points such that
  - (x, y) → (xW, yW, W)
  - (x/W, y/W) are Cartesian coordinates of the homogeneous point (x, y, W)
  - (x, y, 0) represents a point at infinity
  - (0, 0, 0) not allowed

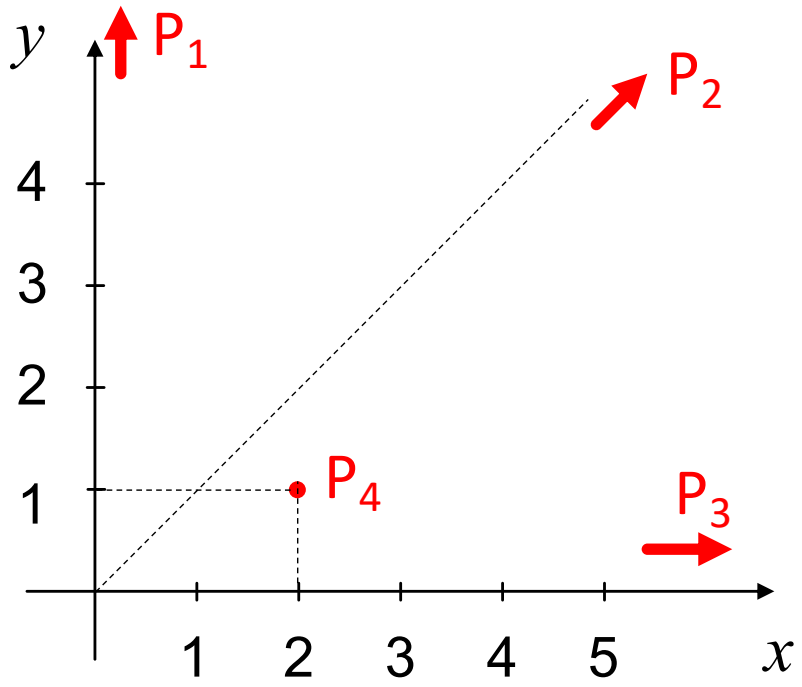




Homogeneous point is a line in 3D space

Homogeneous coordinates

- Even points infinitely far have a representation in homogeneous coordinates
  - Points at infinity have their last coordinate equal to zero
- Examples: P_1=(0,1,0); P_2=(1,1,0); P_3=(1,0,0), P_4=(2,1,1)



Matrix form of elementary 2D transformations

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & T_x \\ 0 & 1 & T_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Translation $\mathcal{T}(T_x,T_y)$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Scaling $\mathcal{S}(S_x,S_y)$

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

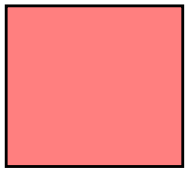
Rotation $\mathcal{R}(\theta)$

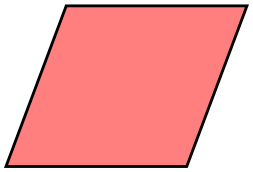
2D Shear transformation

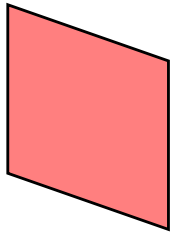
General shear operation

$\mathcal{SH}(Hx, Hy)$

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & H_x & 0 \\ H_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$







Shear in the *X*-direction:

H_y=0

$$SH_x = \begin{bmatrix} 1 & a & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

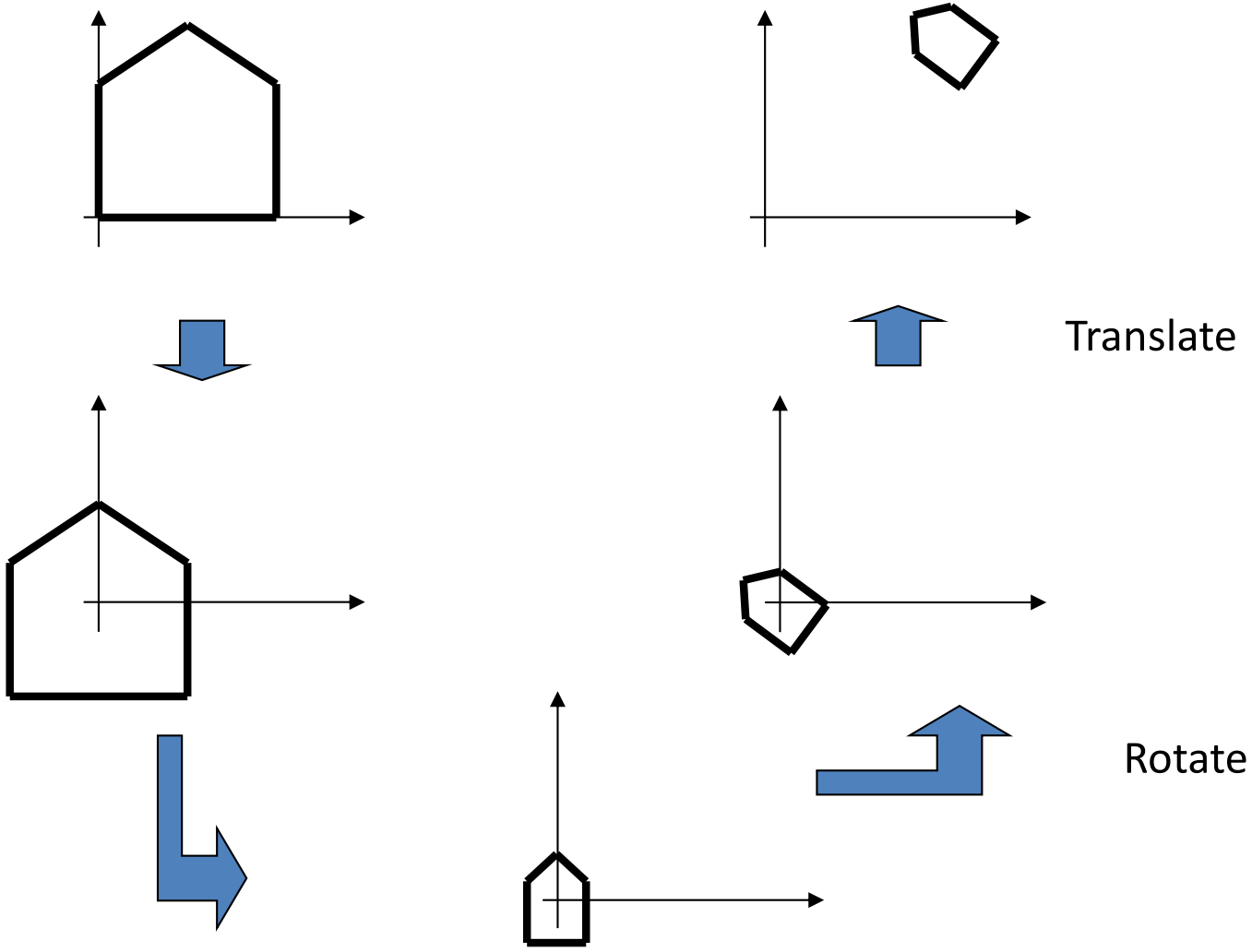
Shear in the *Y*-direction:

H_x=0

$$SH_y = \begin{bmatrix} 1 & 0 & 0 \\ b & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Shear can be represented as a combination of rotations and non-uniform scaling operations.

Compositions of 2D transformations



Example: A complex transformation as a sequence of elementary transformations

Compositions of 2D transformations

- Inverse elementary transforms
  - $\mathcal{T}^{-1}(T_X, T_Y) = \mathcal{T}(-T_X, -T_Y)$
  - $\mathcal{S}^{-1}(S_X, S_Y) = \mathcal{S}(1/S_X, 1/S_Y)$
  - $\mathcal{R}^{-1}(\theta) = \mathcal{R}(-\theta)$
- Complex transformations can be described as a combination (composition) of elementary transformations

$$(X', Y', 1)^T = \mathcal{S}(-)\,\mathcal{R}(-)\,\mathcal{T}(-)\,\mathcal{R}(-)\,\mathcal{S}(-)\,(X, Y, 1)^T$$

- Each rotation brings an extra parameter, scaling and translations two extra parameters
- Can this be simplified?

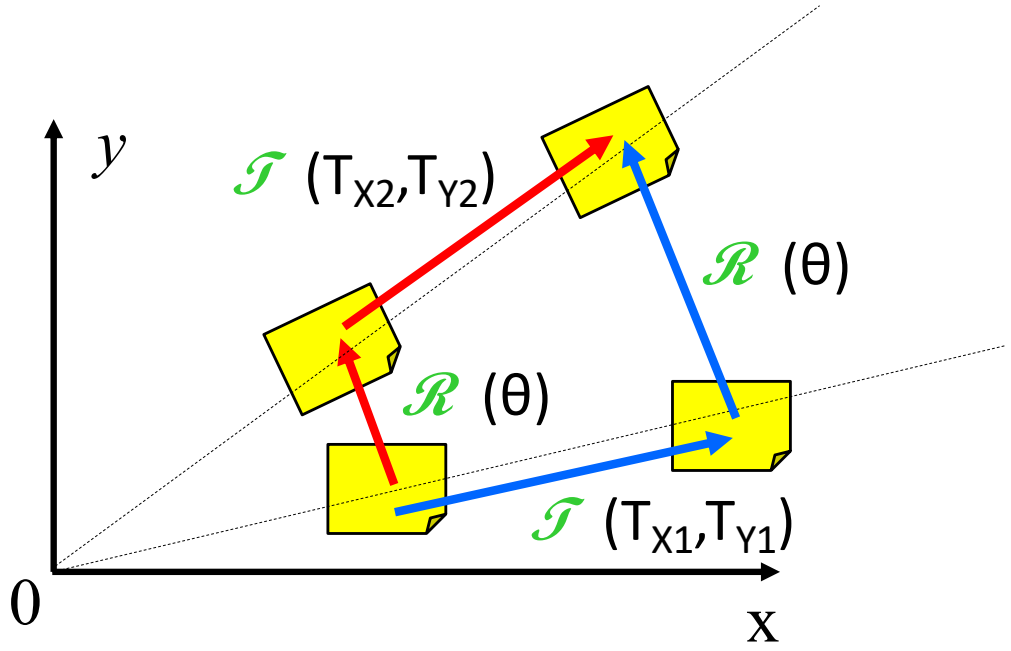
Compositions of 2D transformations

- We try to simplify such expressions by
  - Combining transformations, if possible
  - Changing the order of transformations in order to make the combinations possible
- Sequential elementary transforms of the same type "absorb" each other
  - $\mathcal{T}(T_{X1}, T_{Y1})\,\mathcal{T}(T_{X2}, T_{Y2}) = \mathcal{T}(T_{X1}+T_{X2}, T_{Y1}+T_{Y2})$
  - $\mathcal{S}(S_{X1}, S_{Y1})\,\mathcal{S}(S_{X2}, S_{Y2}) = \mathcal{S}\,(S_{X1}\cdot S_{X2}, S_{Y1}\cdot S_{Y2})$
  - $\mathcal{R}(\theta_1)\,\mathcal{R}(\theta_2) = \mathcal{R}(\theta_1+\theta_2)$
- Changing transformation order is not always possible!
- Transformations that commute are only
  - Two elementary transformations of the same type
  - Scaling with $S_x=S_y$ and Rotation

Compositions of 2D transformations

- Translation and rotation pseudo-commute

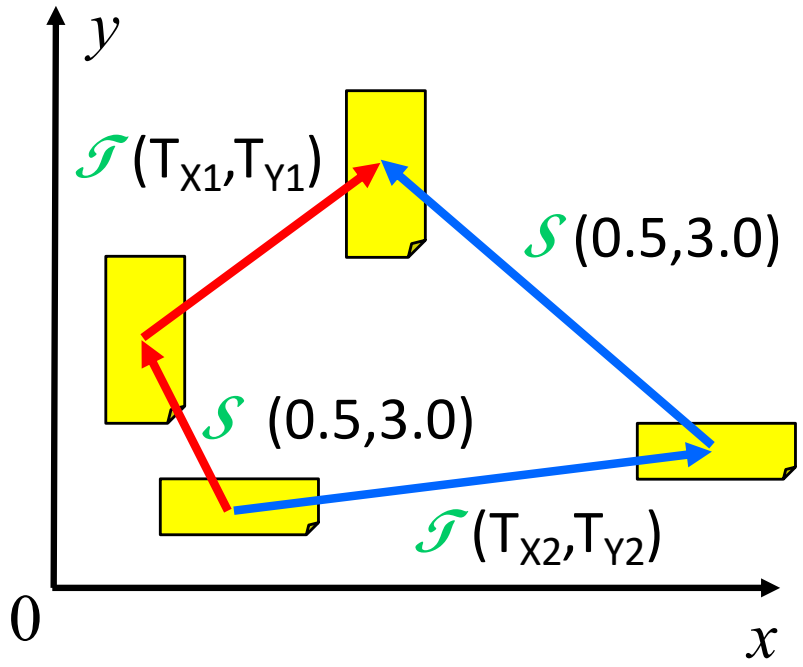
$$\mathcal{T}(T_{X1},T_{Y1})\,\mathcal{R}(\theta) = \mathcal{R}(\theta)\,\mathcal{T}(T_{X2},T_{Y2})$$



Compositions of 2D transformations

- Translation and scaling pseudo-commute

$$\mathcal{S}(S_X, S_Y)\,\mathcal{T}(T_{X1},T_{Y1}) = \mathcal{T}(T_{X2},T_{Y2})\,\mathcal{S}(S_X, S_Y)$$



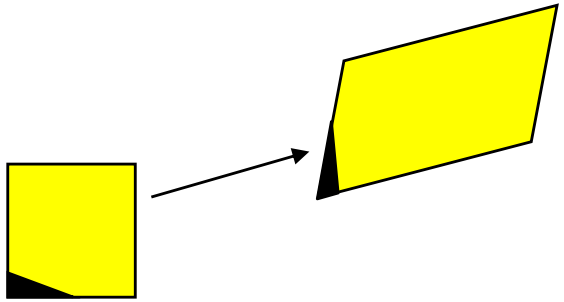
- No similar property for rotation and scaling

Compositions of 2D transformations

- Hypothesis: The most general affine transform can always be represented as

$$\mathcal{R}(\theta_1)\ \mathcal{S}(S_X, S_Y)\ \mathcal{R}(\theta_2)\ \mathcal{T}(T_X, T_Y)$$

- This means: a unit square in the centre is reshaped to an arbitrary parallelogram, brought to an arbitrary position and rotated by an arbitrary angle



$$X' = aX + bY + c$$
$$Y' = dX + eY + f$$

- Prove the hypothesis formally
  - Set $c=T_x$, $f=T_y$
  - Determine the parameters θ_1, θ_2, S_X, S_Y as a function of a, b, d, e

Efficiency: matrix calculations

- The most general affine transformation

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- Multiplication of a 3x3 matrix with column vector requires 9 multiplications and 6 additions

- Actually we need only 4 multiplications and 4 additions
  - X’ = aX + bY + c
  - Y’ = dX + eY + f

- Even though matrix representation is useful, practical application should make use of the special structure of the matrix, for efficiency

Efficiency: creating successive views

- To produce an impression of a dynamically rotating object, many successive views are needed
  - $x' = x\cos\theta - y\sin\theta$
  - $y' = x\sin\theta + y\cos\theta$
- The angle difference between the successive views is very small (a few degrees). Can we simplify the calculation?
- Solution 1: Use approximation $\cos\theta \approx 1$:
  - $x' = x - y\sin\theta$
  - $y' = x\sin\theta + y$
  - What is wrong with this solution?
- A better solution
  - $x' = x - y\sin\theta$
  - $y' = x'\sin\theta + y$ Check the determinant of the matrix in both cases!

3D transformations

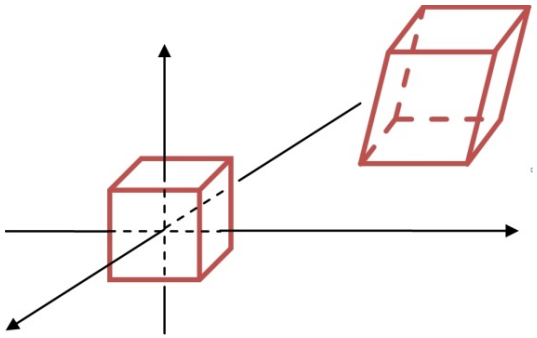
3-D Transformations

- Generalization of 2D transformations
- Same principle: apply to vertices (which are now 3D points)
  - P(X,Y,Z) is transformed to P'(X',Y',Z')
- We consider general 3D affine transformation

$$X' = aX + bY + cZ + d$$

$$Y' = eX + fY + gZ + h$$

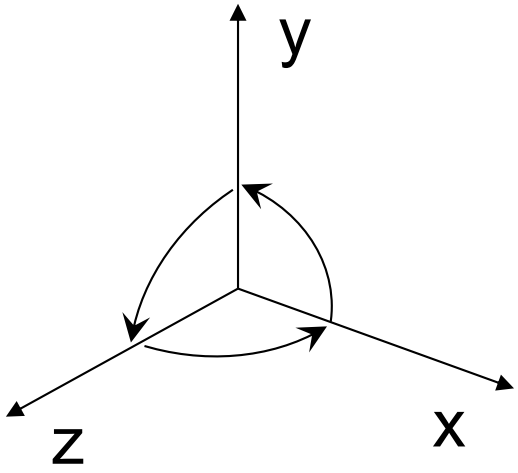
$$Z' = iX + jY + kZ + l$$

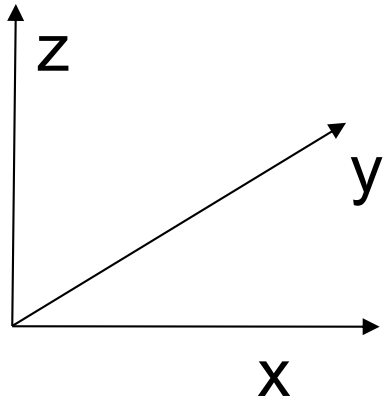


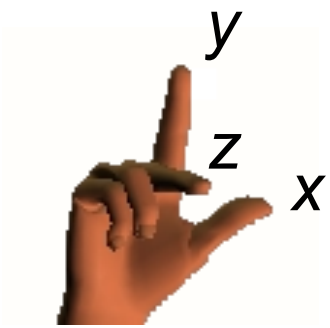
- Properties analogous to 2D case
  - Lines map to lines (plane segments map to plane segments)
  - Parallelism preserved
  - A unit cube centered in the origin is transformed into an arbitrary **parallelepiped**, arbitrarily positioned in space

3D transformations

- Convention that we will adopt:
  - Right-hand side system

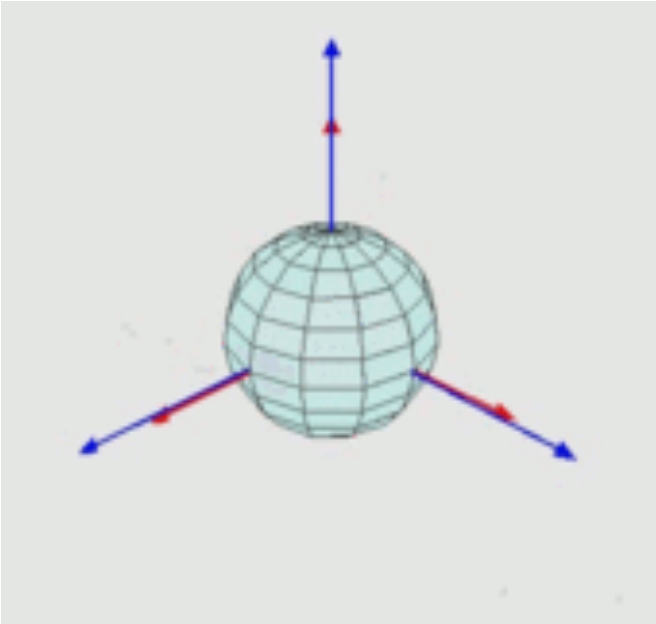


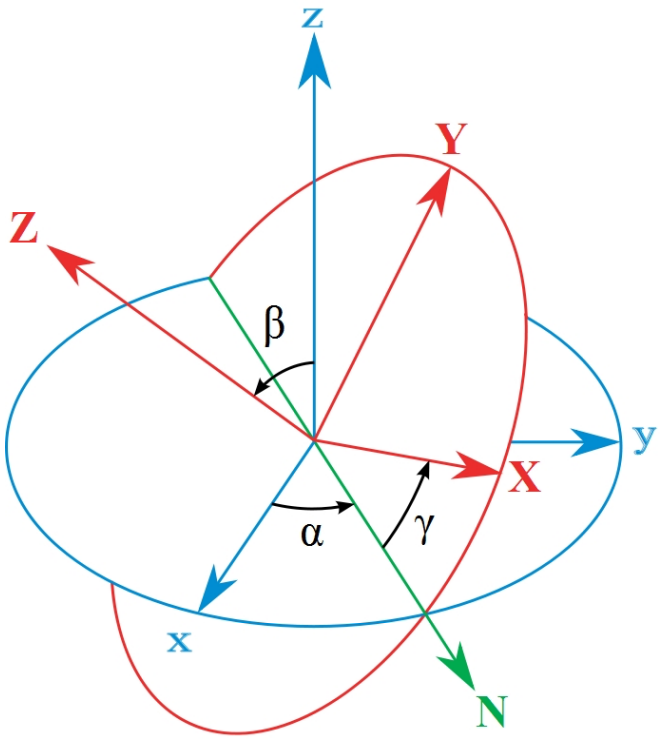




3D rotation

- 3D rotation of a rigid object is described by three parameters, such as three angles of Euler
- In Euler angle formulation an arbitrary rotation is represented as a composition of three elemental rotations, each around a single coordinate axis

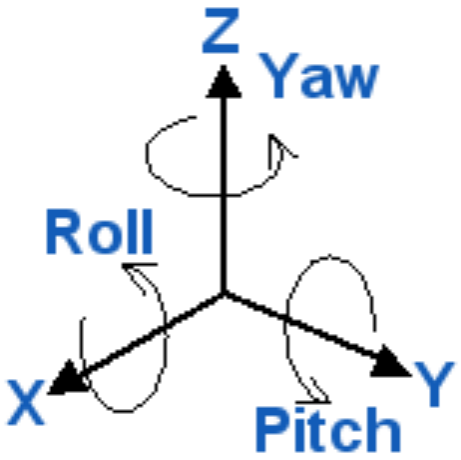


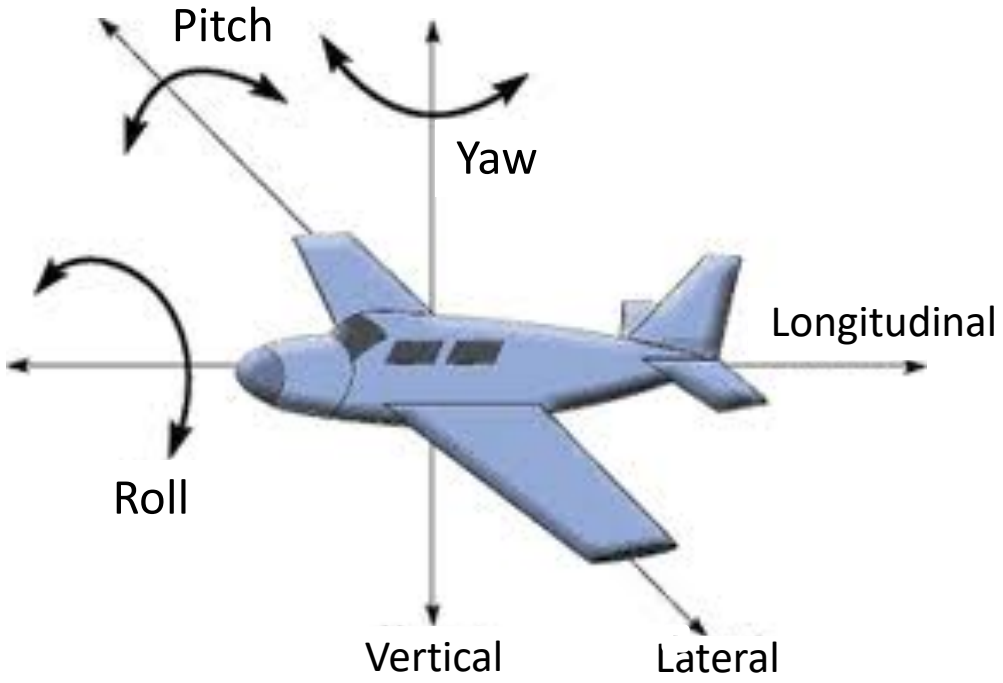


http://en.wikipedia.org/wiki/Euler_angles

3D rotation

- The rotations around x, y and z axis are also known as Roll, Pitch and Yaw (“heading”); terms coming from flight dynamics





3D transformations

- Three elementary transforms
  - Elementary translation
  - Three elementary rotations
    - Around the X-axis, Y-axis and Z-axis
  - Elementary scaling
    - In X-, Y-, en Z-direction

- Matrix representation
  - 4 x 4 matrices for 3-D, analogous to 3 x 3 matrices for 2-D using homogeneous coordinates

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} \rightarrow \begin{bmatrix} Wx \\ Wy \\ Wz \\ W \end{bmatrix}; \quad \text{usual notation } W = 1: \quad \begin{bmatrix} x \\ y \\ z \end{bmatrix} \rightarrow \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Each point in 3D space → a line throught the origin in 4D space

3D Translation and Scaling

- Translation

$X' = X + T_X$

$Y' = Y + T_Y$

$Z' = Z + T_Z$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\mathcal{T}(T_X, T_Y, T_Z)} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

- Elementary scaling

$X' = S_X X$

$Y' = S_Y Y$

$Z' = S_Z Z$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} S_X & 0 & 0 & 0 \\ 0 & S_Y & 0 & 0 \\ 0 & 0 & S_Y & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\mathcal{S}(S_X, S_Y, S_Z)} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

3D Rotation

- Rotation around x-axis

X’ = X
Y’ = Y cos(θ) - Z sin(θ)
Z’ = Y sin(θ) + Z cos(θ

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

- Similarly for rotations around other two axes:

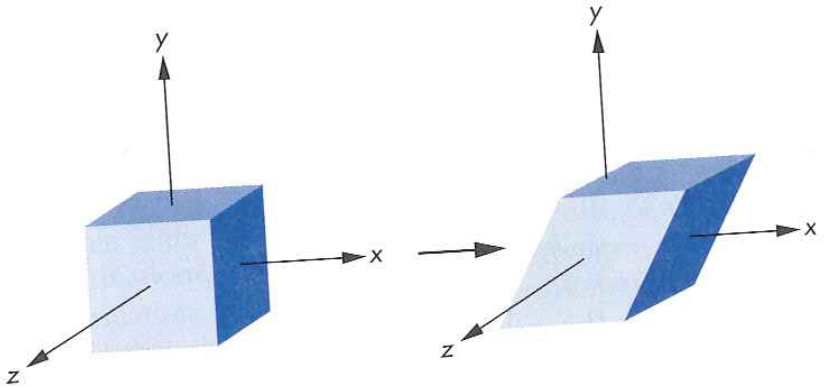
$$\mathcal{R}_Y(\varphi) = \begin{bmatrix} \cos\varphi & 0 & \sin\varphi & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\varphi & 0 & \cos\varphi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathcal{R}_Z(\psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 & 0 \\ \sin\psi & \cos\psi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3D Shear

- Shear around the Z-axis

$$X' = X + H_X Z$$
$$Y' = Y + H_Y Z$$
$$Z' = Z$$



- Can be represented in matrix form as

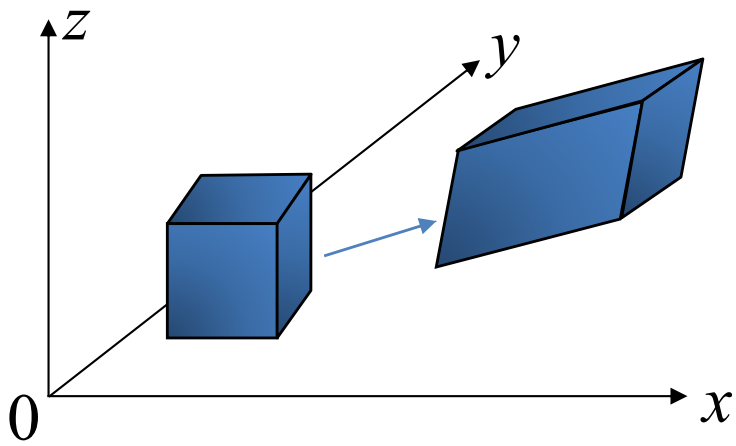
$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & H_x & 0 \\ 0 & 1 & H_y & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

- Shear around the X-axis and Y-axis have a similar form

3D Transformations: properties

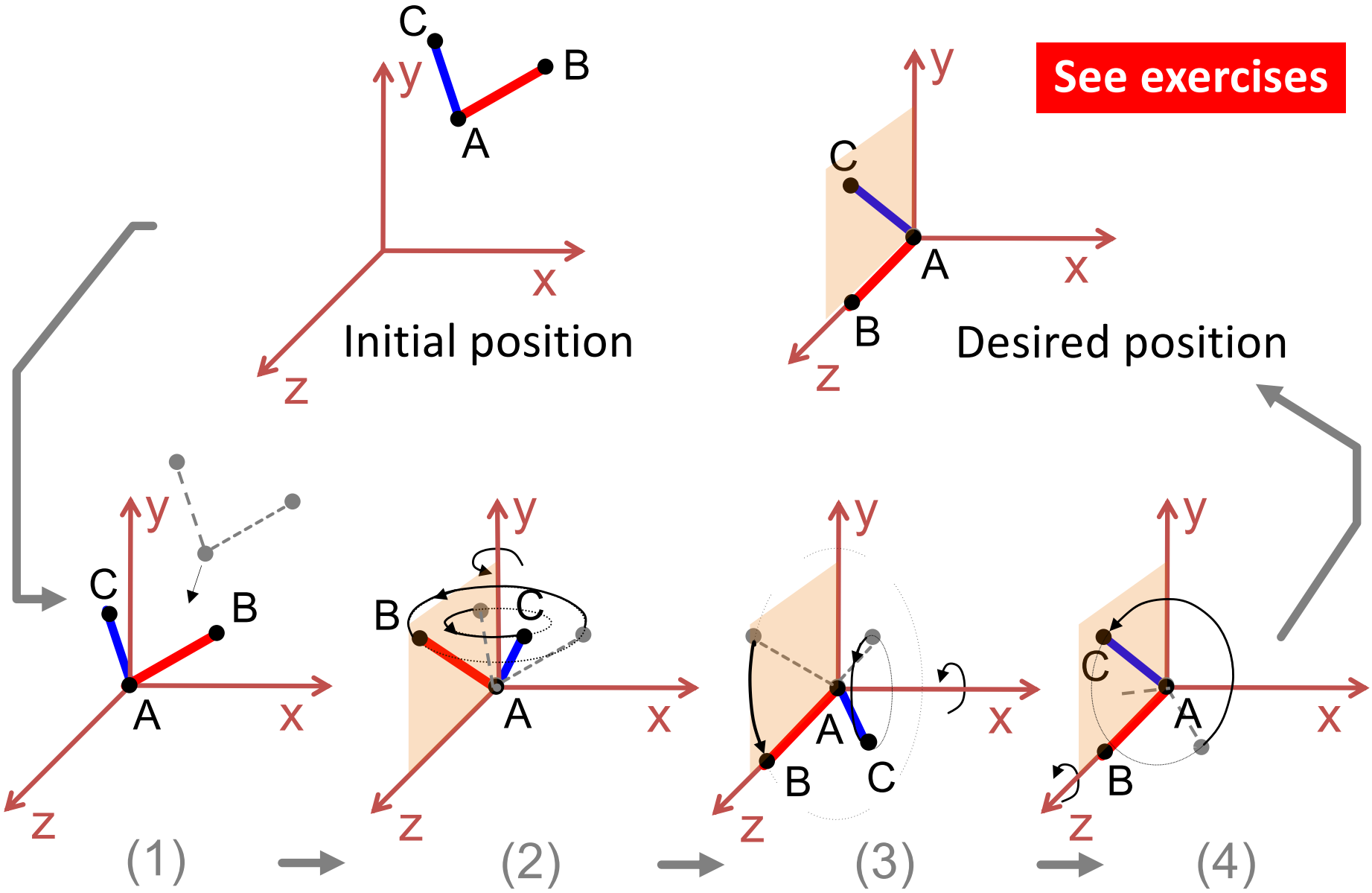
- Inverse transforms are defined as
  - $\mathcal{T}^{-1}(T_X, T_Y, T_Z) = \mathcal{T}(-T_X, -T_Y, -T_Z)$
  - $\mathcal{S}^{-1}(S_X, S_Y, S_Z) = \mathcal{S}(1/S_X, 1/S_Y, 1/S_Z)$
  - $\mathcal{R}_X^{-1}(\theta) = \mathcal{R}_X(-\theta)$; $\mathcal{R}_Y^{-1}(\varphi) = \mathcal{R}_Y(-\varphi)$; $\mathcal{R}_Z^{-1}(\psi) = \mathcal{R}_X(-\psi)$
- The same "absorption" and "pseudo-commutation" are valid like in the 2D case
  - Translation pseudo-commutes with scaling and with three rotations
  - Scaling and rotations do not pseudo-commute
- Additional properties, specific for 3D (without proof)
  - Two (different) rotations do not pseudo-commute (not even with an additional translation)
  - Three (different) rotations and a translation do pseudo commute

3D Transformations: properties



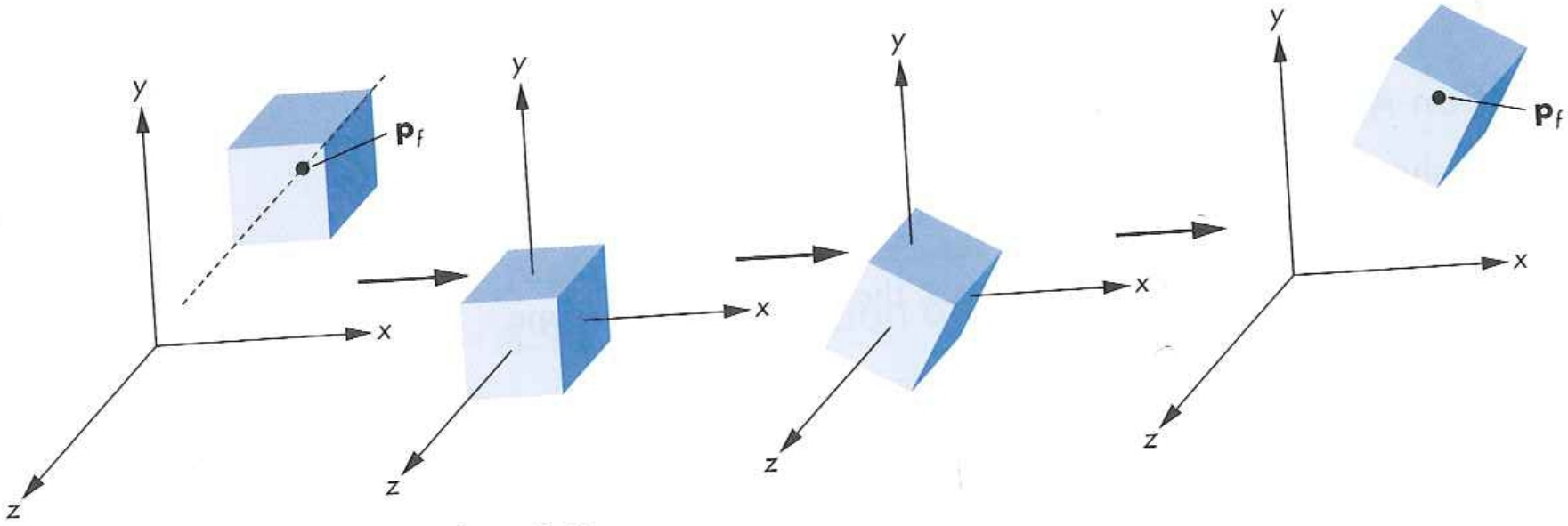
- A general positioning transformation involves 6 parameters
  - $\mathcal{T}(T_X, T_Y, T_Z)\ \mathcal{R}_X(\theta)\ \mathcal{R}_Y(\varphi)\ \mathcal{R}_Z(\psi)$
- Object deformation: 6 other parameters (scaling + rotation)
- A general affine transformation involves thus 12 parameters
  - $\mathcal{R}_X(\theta)\ \mathcal{R}_Y(\varphi)\ \mathcal{R}_Z(\psi)\ \mathcal{S}(S_X,S_Y,S_Z)\ \mathcal{R}_X(\theta')\ \mathcal{R}_Y(\varphi')\ \mathcal{R}_Z(\psi')\ \mathcal{T}(T_X,T_Y,T_Z)$
- There are many equivalent forms

Arbitrary displacement in 3D



Compositions of 3D transformations

Rotation about arbitrary point: in this example about z-axis



From the book of Edward Angel: Interactive Computer Graphics – A Top-Down Approach Using OpenGL

Quaternions and 3D rotation

Complex numbers and rotation

- Polar representation of a complex number

$$c = a + \mathbf{i}\,b = re^{\mathbf{i}\theta} \qquad r = \sqrt{a^2 + b^2}, \quad \theta = \arctan(b/a)$$

- Denote by c' the result of rotating c about the origin by ϕ:

$$c' = re^{\mathbf{i}\theta}e^{\mathbf{i}\phi} = re^{\mathbf{i}(\theta+\phi)}$$

- Quaternions (Sir William Rowan Hamilton, 1843) are extensions of complex numbers in 3D, yielding elegant rotations in 3D

$$a = (q_0, \mathbf{q}); \qquad \mathbf{q} = q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k}$$

$$\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -1$$

Complex numbers and rotation

- A point in space $p = (0, \mathbf{p})$
- Rotation of the point p by θ about the unit-length vector $\mathbf{v}$:

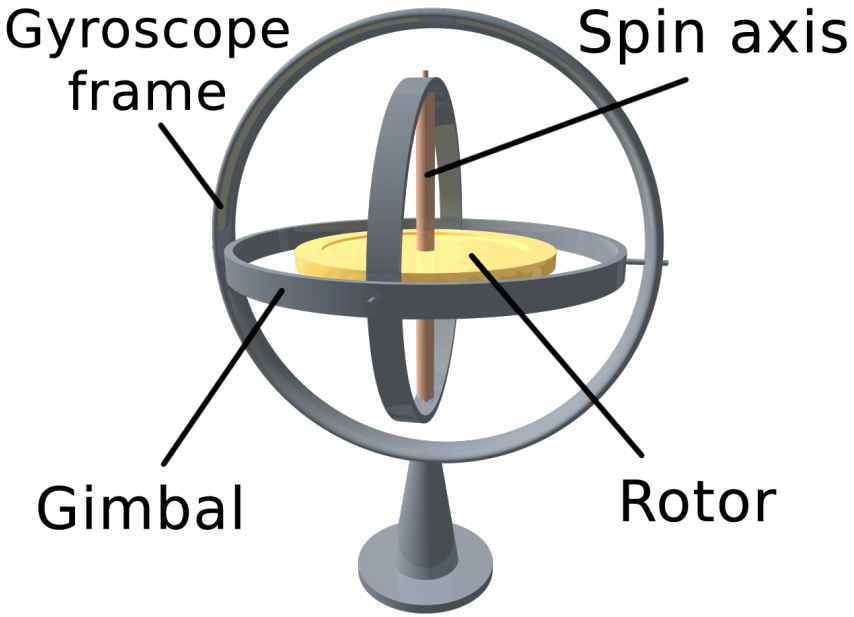
$$p' = rpr^{-1}$$

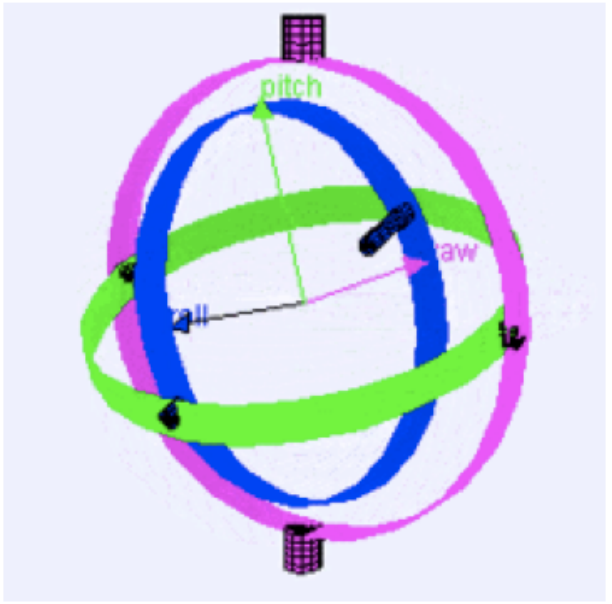
where:

$$r = \left(\cos\frac{\theta}{2}, \sin\frac{\theta}{2}\mathbf{v} \right), \qquad r^{-1} = \left(\cos\frac{\theta}{2}, -\sin\frac{\theta}{2}\mathbf{v} \right)$$

- Why are quaternions interesting for 3D rotations in computer graphics?
  - Euler angle representation suffers from Gimbal lock
  - Simpler representation
  - Lower computational complexity, but not necessarily after conversion to matrix form

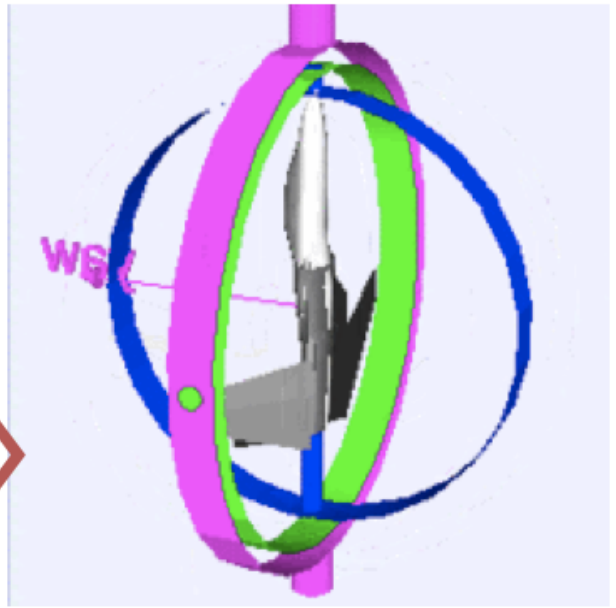
Gimbal lock





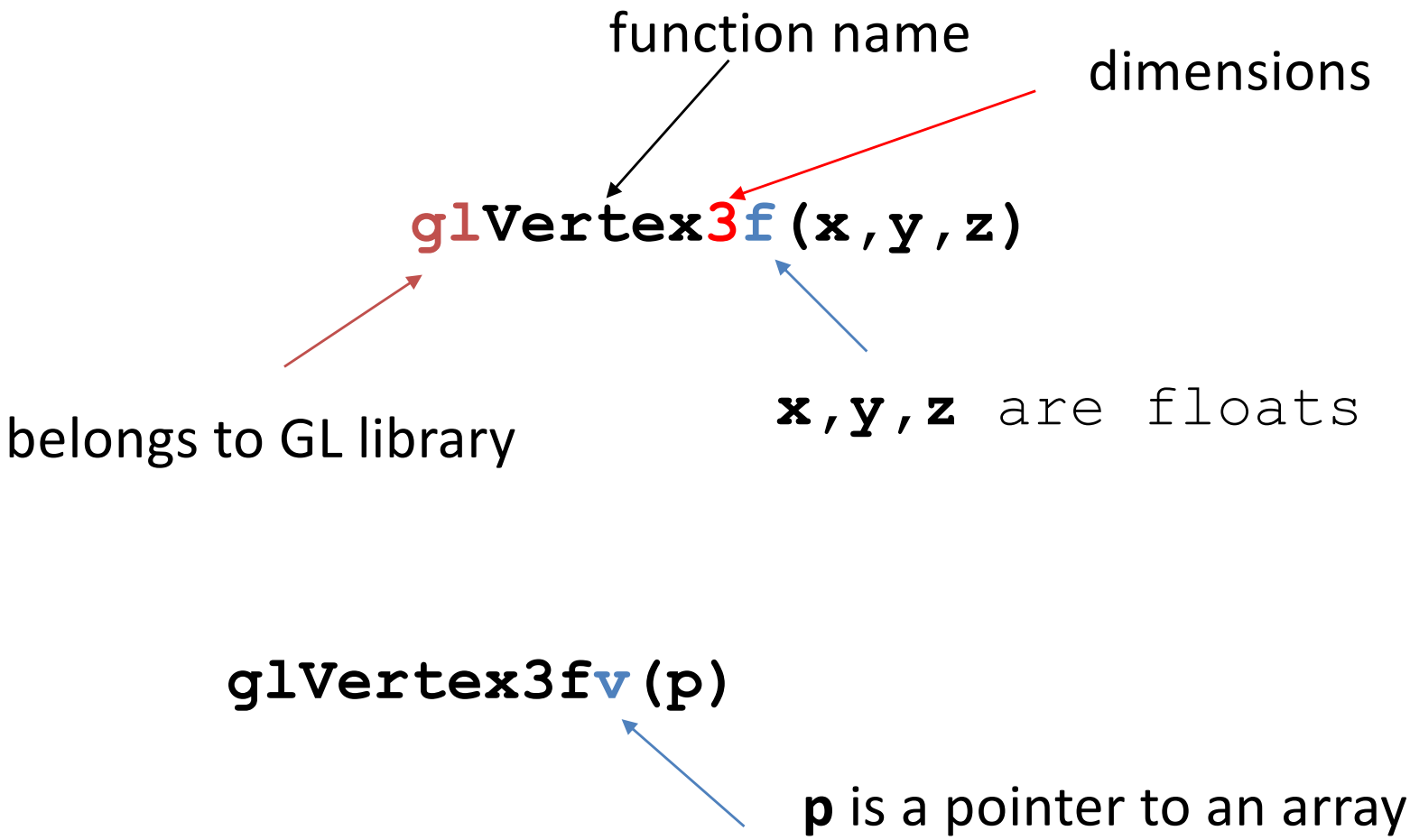
When the axes of two of the three gimbals align, "locking" into rotation in a degenerate 2D space.

Example: when the pitch (green) and yaw (magenta) gimbals become aligned, changes to roll (blue) and yaw apply the same rotation to the airplane.



Transformations in OpenGL

OpenGL function format



Transformations in OpenGL

- We can load matrix with the function

```
glLoadMatrix(pointer_to_matrix)
```

- Or set a matrix to identity matrix with the function

```
glLoadIdentity()
```

- Rotation, translation and scaling are provided through

```
glRotatef(angle,vx,vy,vz)
glTranslatef(dx,dy,dz)
glScalef(sx,sy,sz)
```

Summary

- 2D transformations
  - arbitrary 2D affine transformation 6 parameters
  - complex transformations as compositions of the elementary ones
- 3D transformations
  - arbitrary 3D affine transformation 12 parameters
  - complex transformations as compositions of the elementary ones
  - Euler angles
- Quaternions
  - advantage over Euler angle representation (Gimbal lock avoided)
- Transformations in OpenGL