Computer Graphics

- Transformations -

Alexander Rath
Philipp Slusallek
Slides by Philipp Slusallek
and Piotr Danilewski

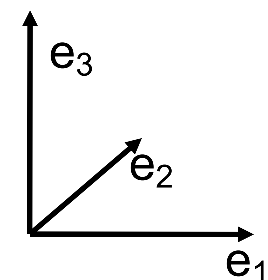
Overview

- **Last time**
  - Introduction to Ray Tracing
- **Today**
  - Vector spaces and affine spaces
  - Homogeneous coordinates
  - Basic transformations in homogeneous coordinates
  - Concatenation of transformations
  - Projective transformations

Vector Space

- **Math recap**
  - 3D vector space over the real numbers
    - $\boldsymbol{v} = \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix} \in \boldsymbol{V}^{\mathbf{3}} = \mathbb{R}^{\mathbf{3}}$
  - Vectors written as n x 1 matrices
  - Vectors describe directions – **not positions**!
    - All vectors start from the origin of the coordinate system
  - 3 linear independent vectors create a basis
    - Standard basis

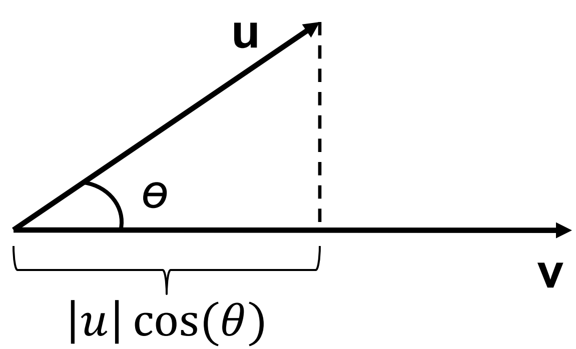
$$\{\boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_3\} = \left\{ \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \right\}$$



  - Any vector can now be represented uniquely with coordinates v_i
    - $\boldsymbol{v} = v_1\boldsymbol{e_1} + v_2\boldsymbol{e_2} + v_3\boldsymbol{e_3} \quad v_1, v_2, v_3 \in \mathbb{R}$

Vector Space - Metric

- **Standard scalar product a.k.a. dot or inner product**
  - Measure lengths
    - $|v|^2 = v \cdot v = v_1^2 + v_2^2 + v_3^2$
  - Compute angles
    - $u \cdot v = |u||v| \cos(u, v)$
  - Projection of vectors onto other vectors
    - $|u| \cos(\theta) = \frac{u \cdot v}{|v|} = \frac{u \cdot v}{\sqrt{v \cdot v}}$

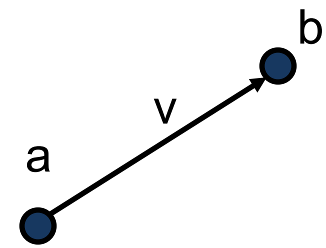


Vector Space - Basis

- **Orthonormal basis**
  - Unit length vectors
    - $|e_1| = |e_1|$= $|e_1|$=1
  - Orthogonal to each other
    - $e_i \cdot e_j = \delta_{ij}$
- **Handedness of the coordinate system**
  - $e_1 \times e_2 = \pm e_3$
    - Positive: Right-handed
    - Negative: Left-handed

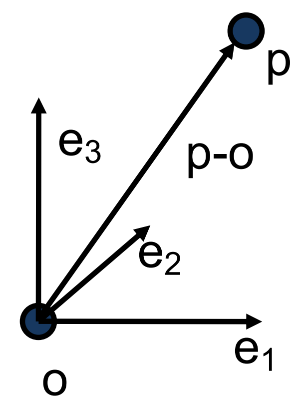
Affine Space

- **Basic mathematical concepts**
  - Denoted as A^3
    - Elements are positions (not directions)
  - Defined via its associated vector space V^3
    - $a, b \in A^3 \Leftrightarrow \exists!\, v \in V^3 : v = b - a$
    - $\rightarrow$: unique, $\leftarrow$: ambiguous
  - Operations on A^3
    - Subtraction yields a vector
    - No addition of affine elements
      - Its not clear what the some of to points would mean
    - But: Addition of points and vectors:
      - $a + v = b \in A^3$
    - Distance
      - $dist(a, b) = |a - b|$



Affine Space - Basis

- **Affine Basis**
  - Given by its origin **o** (a point) and the basis of an associated vector space
    - $\{\boldsymbol{e_1}, \boldsymbol{e_2}, \boldsymbol{e_3}, \boldsymbol{o}\}$: $\boldsymbol{e_1}, \boldsymbol{e_2}, \boldsymbol{e_3} \in V^3; \boldsymbol{o} \in \boldsymbol{A^3}$
- **Position vector of point p**
  - $(p - o)$ is in V^3

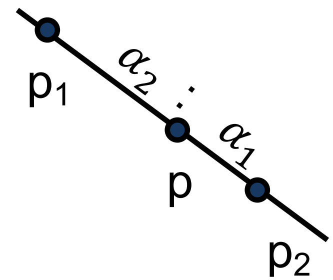


Affine Coordinates

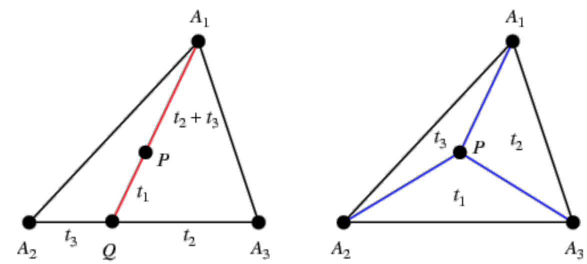
- **Affine Combination**
  - Linear combination of (n+1) points
    - $p_0, \dots, p_n \in A^n$
  - With weights forming a partition of unity
    - $\alpha_0, \dots, \alpha_n \in \mathbb{R}$ with $\sum_i \alpha_i = 1$
  - $p = \sum_{i=0}^{n} \alpha_i p_i = p_0 + \sum_{i=1}^{n} \alpha_i (p_i - p_0) = o + \sum_{i=1}^{n} \alpha_i v_i$
- **Basis**
  - $(n + 1)$ points form am **affine basis** of A^n
    - Iff none of these point can be expressed as an affine combination of the other points
    - Any point in A^n can then be uniquely be represented as an affine combination of the affine basis $p_0, \dots, p_n \in A^n$
    - Any vector in another basis can be expressed as a linear combination of the p_i, yielding a matrix for the basis

Affine Coordinates

- **Closely related to "Barycentric Coordinates"**
  - Center of mass of $(n+1)$ points with arbitrary masses (weights) m_i is given as
    - $p = \frac{\sum m_i p_i}{\sum m_i} = \sum \frac{m_i}{\sum m_i} p_i = \sum \alpha_i p_i$

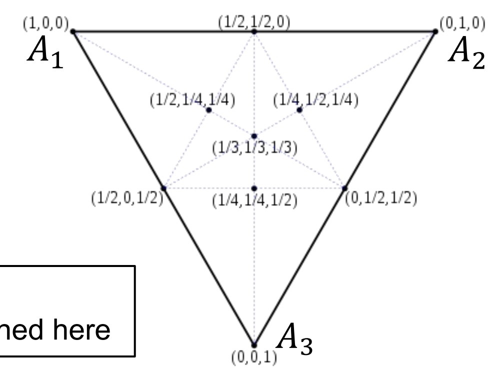


- **Convex / Affine Hull**
  - If all α_i are non-negative than p is in the **convex hull** of the other points
- **In 1D**
  - Point is defined by the splitting ratio $\alpha_1 \colon \alpha_2$
    - $p = \alpha_1 p_1 + \alpha_2 p_2 = \frac{|p-p_2|}{|p_2-p_1|} p_1 + \frac{|p-p_1|}{|p_2-p_1|} p_2$



- **In 2D**
  - Weights are the relative areas in $\Delta(A_1, A_2, A_3)$
    - $t_i = \alpha_i = \frac{\Delta(P, A_{(i+1)\%3}, A_{(i+2)\%3})}{\Delta(A_1, A_2, A_3)}$
    - $p = \alpha_1 A_1 + \alpha_2 A_2 + \alpha_3 A_3$

Note: Length and area measures need to be signed here



Affine Mappings

- **Properties**
  - Affine mapping (continuous, bijective, invertible)
    - T: $A^3 \rightarrow A^3$
  - Defined by two non-degenerated simplicies
    - 2D: Triangle, 3D: Tetrahedron, ...
  - Invariants under affine transformations:
    - Barycentric/affine coordinates
    - Straight lines, parallelism, splitting ratios, surface/volume ratios
  - Characterization via fixed points and lines
    - Given as eigenvalues and eigenvectors of the mapping
- **Representation**
  - Matrix product and a translation vector:
    - $\boldsymbol{T}p = \boldsymbol{A}p + \boldsymbol{t}$ with $\mathbf{A} \in \mathbb{R}^{n \times n}$, $\mathrm{t} \in \mathbb{R}^n$
  - Invariance of affine coordinates
    - $\boldsymbol{T}p = \boldsymbol{T}(\sum \alpha_i p_i) = \boldsymbol{A}(\sum \alpha_i p_i) + \boldsymbol{t} = \sum \alpha_i (\boldsymbol{A}p_i) + \sum \alpha_i \boldsymbol{t} = \sum \alpha_i (\boldsymbol{T}p_i)$

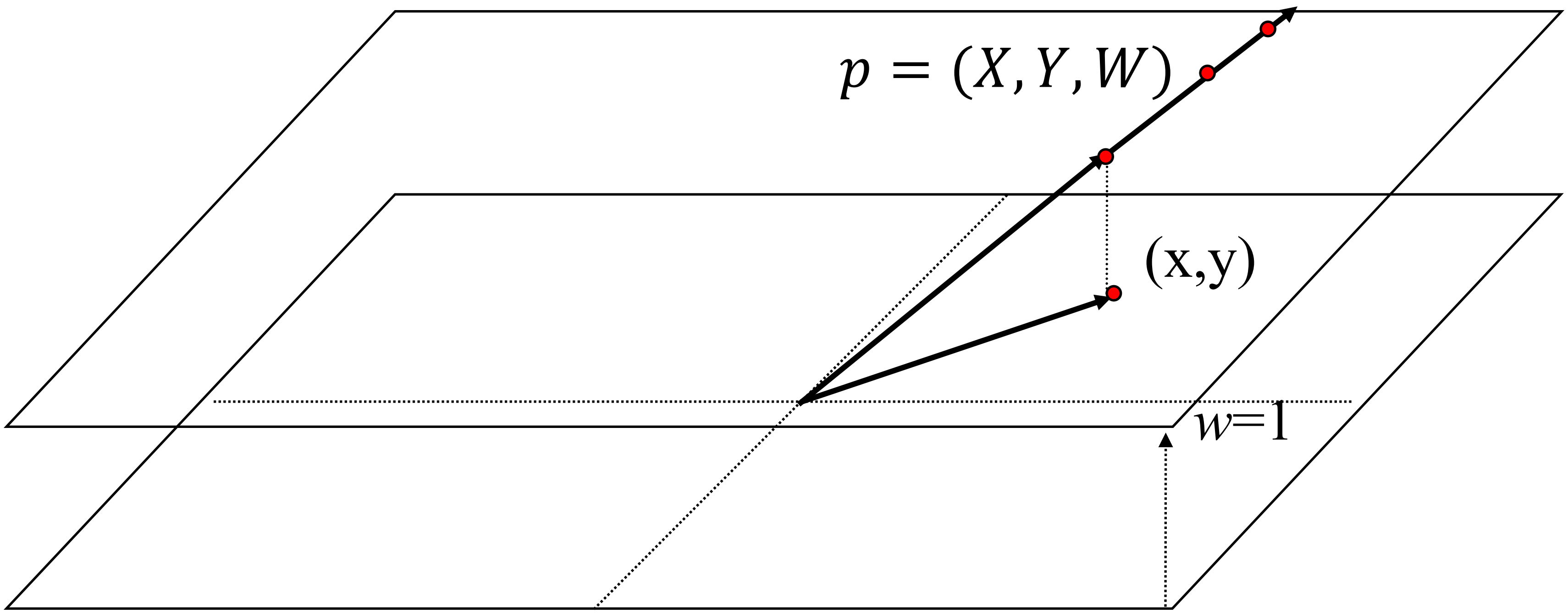
Homogeneous Coordinates for 3D

- **Homogeneous embedding of R^3 into the projective 4D space $P(R^4)$**
  - Mapping into homogeneous space
    - $\mathbb{R}^3 \ni \begin{pmatrix} x \\ y \\ z \end{pmatrix} \longrightarrow \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \in P(\mathbb{R}^4)$
  - Mapping back by dividing through fourth component
    - $\begin{pmatrix} X \\ Y \\ Z \\ W \end{pmatrix} \longrightarrow \begin{pmatrix} X/W \\ Y/W \\ Z/W \end{pmatrix}$

- **Consequence**
  - This allows to represent affine transformations as 4x4 matrices
  - Mathematical trick
    - Convenient representation to express rotations *and* translations as matrix multiplications
    - Easy to find line through points, point-line/line-line intersections
  - Also important for projections (later)

Point Representation in 2D

- **Point in homogeneous coordinates**
  - All points along a line through the origin map to the same point in 2D



$$x = \frac{X}{W} \qquad y = \frac{Y}{W}$$

Homogeneous Coordinates in 2D

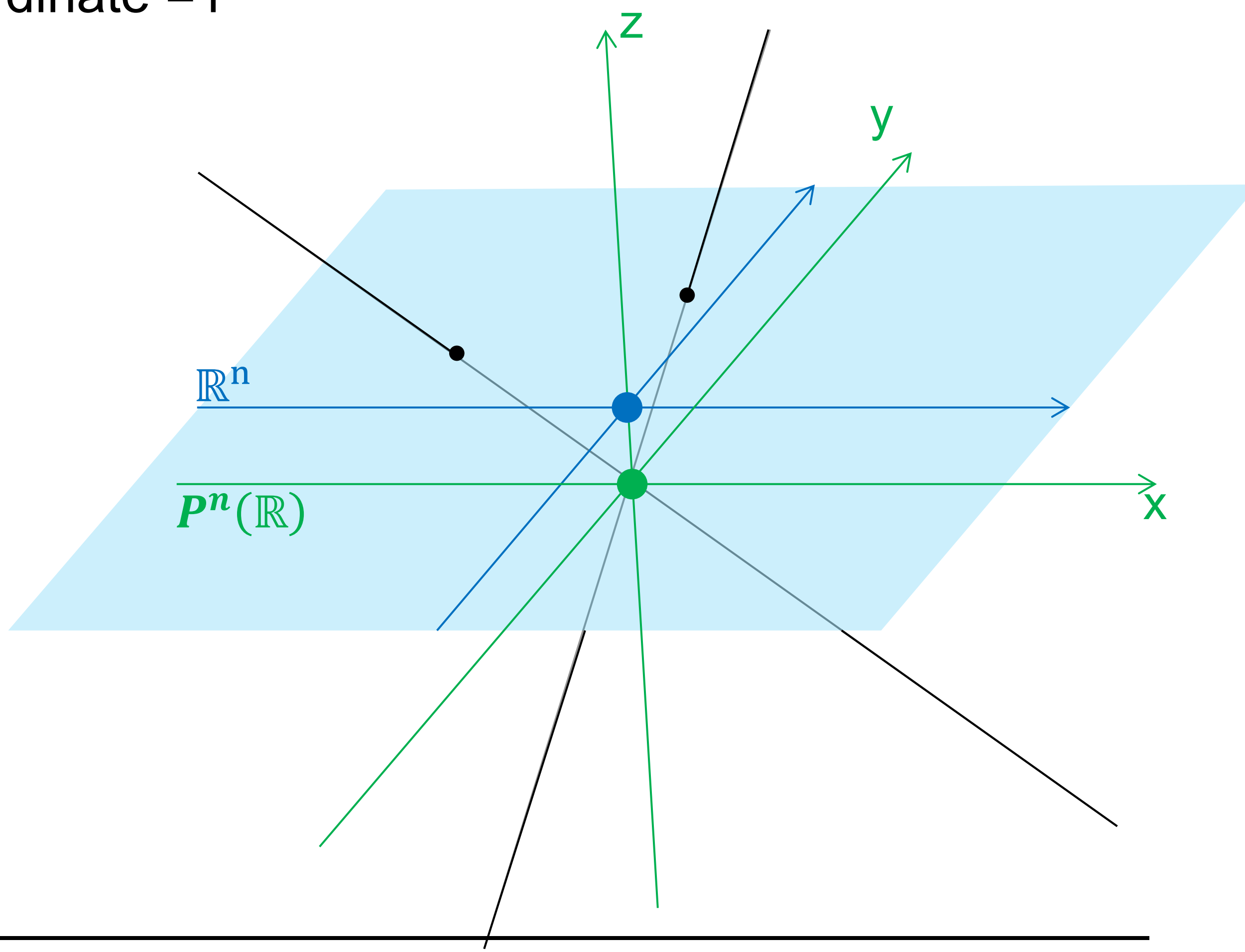
- **Some tricks (works only in P(R³), i.e. only in 2D)**
  - Point representation
    - $(X) = \begin{pmatrix} X \\ Y \\ W \end{pmatrix} \in P(\mathbb{R}^3)$, $\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} X/W \\ Y/W \end{pmatrix}$
  - Representation of a line $l \in \mathbb{R}^2$
    - Dot product of l vector with point in plane must be zero:
      - $l = \left\{ \begin{pmatrix} x \\ y \end{pmatrix} \middle| \, ax + by + c \cdot 1 = 0 \right\} = \left\{ X \in P(\mathbb{R}^3) | X \cdot l = 0, \mathrm{l}= \begin{pmatrix} a \\ b \\ c \end{pmatrix} \right\}$
    - Line l is normal vector of the plane through origin and points on line
  - Intersection of lines l and l':
    - Point on both lines ➔ point must be orthogonal to both line vectors
    - $X \in l \cap l' \Leftrightarrow X = l \times l'$
  - Line trough 2 points p and p'
    - Line must be orthogonal to both points
    - $p \in l \ \wedge \ p' \in l \Leftrightarrow l = p \times p'$

Affine view

- $P^n(\mathbb{R})$ - projective space
- $\mathbb{R}^n$ - affine view
  - typically: last coordinate =1

$P^n(\mathbb{R})$ $\mathbb{R}^n$

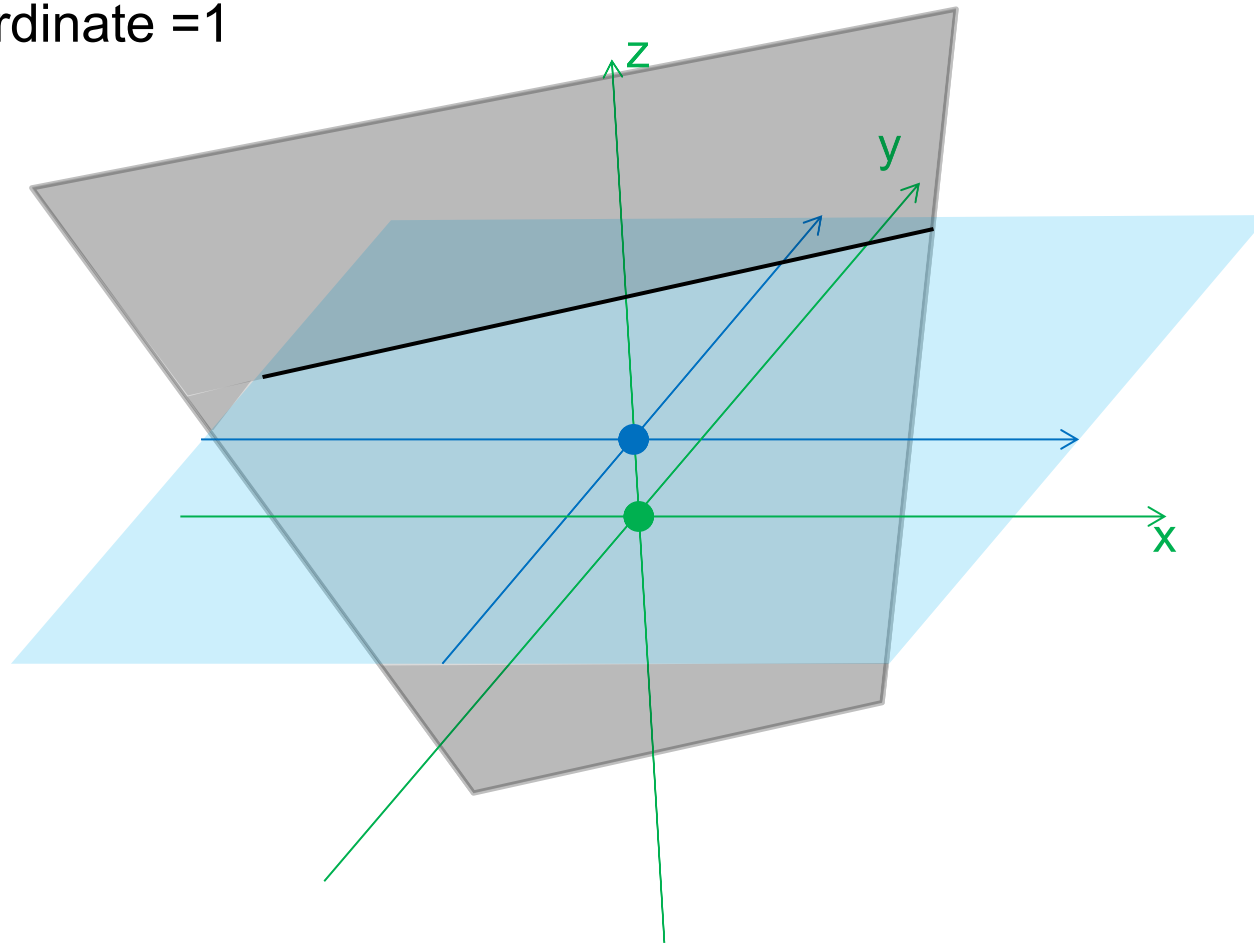
lines $\longrightarrow$ points



Affine view

- $\boldsymbol{P^n}(\mathbb{R})$ - projective space
- $\mathbb{R}^{\mathrm{n}}$ - affine view
  - typically: last coordinate =1

$\boldsymbol{P^n}(\mathbb{R})$		$\mathbb{R}^{\mathrm{n}}$
lines	$\longrightarrow$	points
planes	$\longrightarrow$	lines

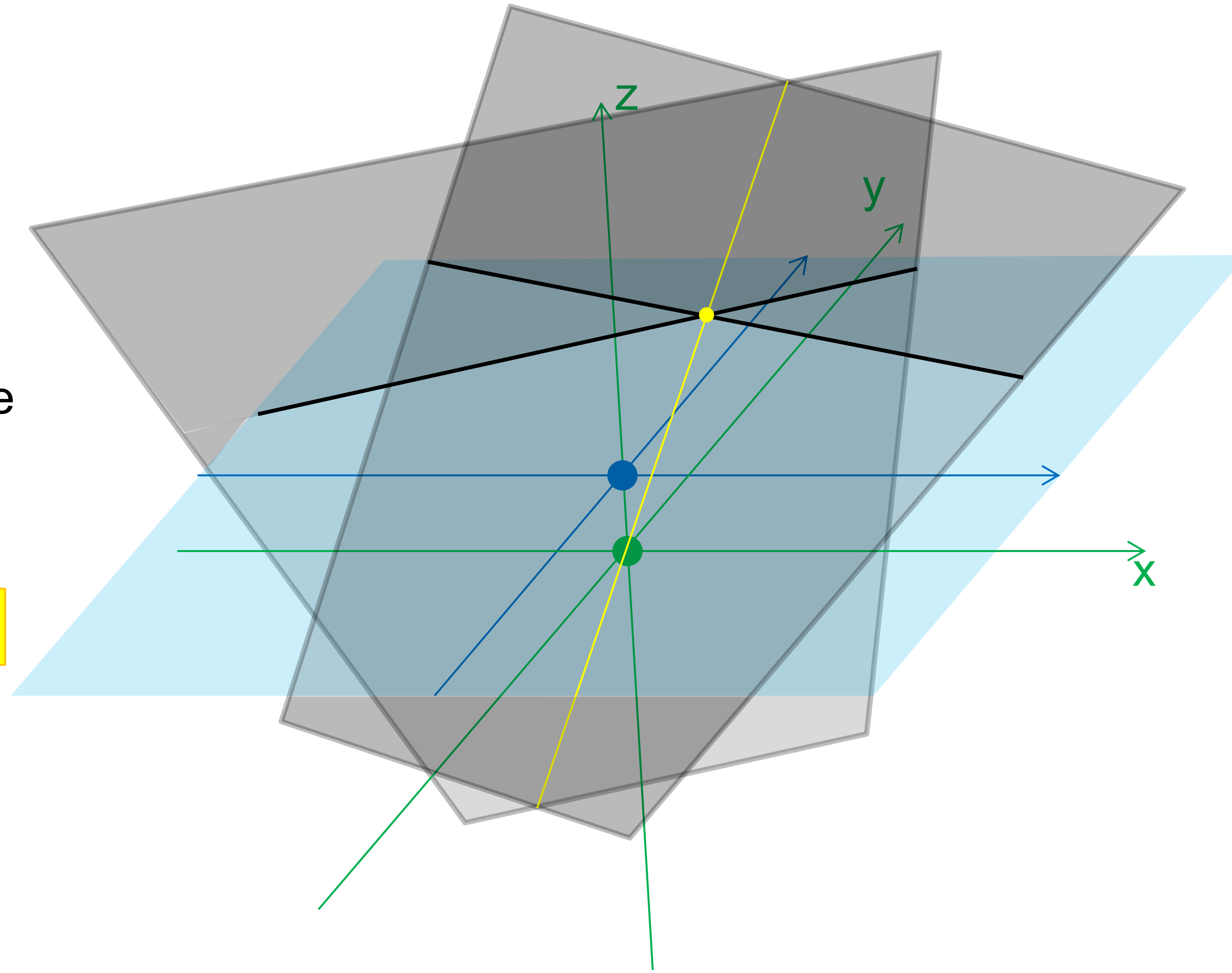


Intersections

- $\boldsymbol{P^n}(\mathbb{R})$ - projective space
- $\mathbb{R}^n$ - affine view

$$
\begin{array}{cc}
\boldsymbol{P^n}(\mathbb{R}) & \mathbb{R}^n \\
\end{array}
$$

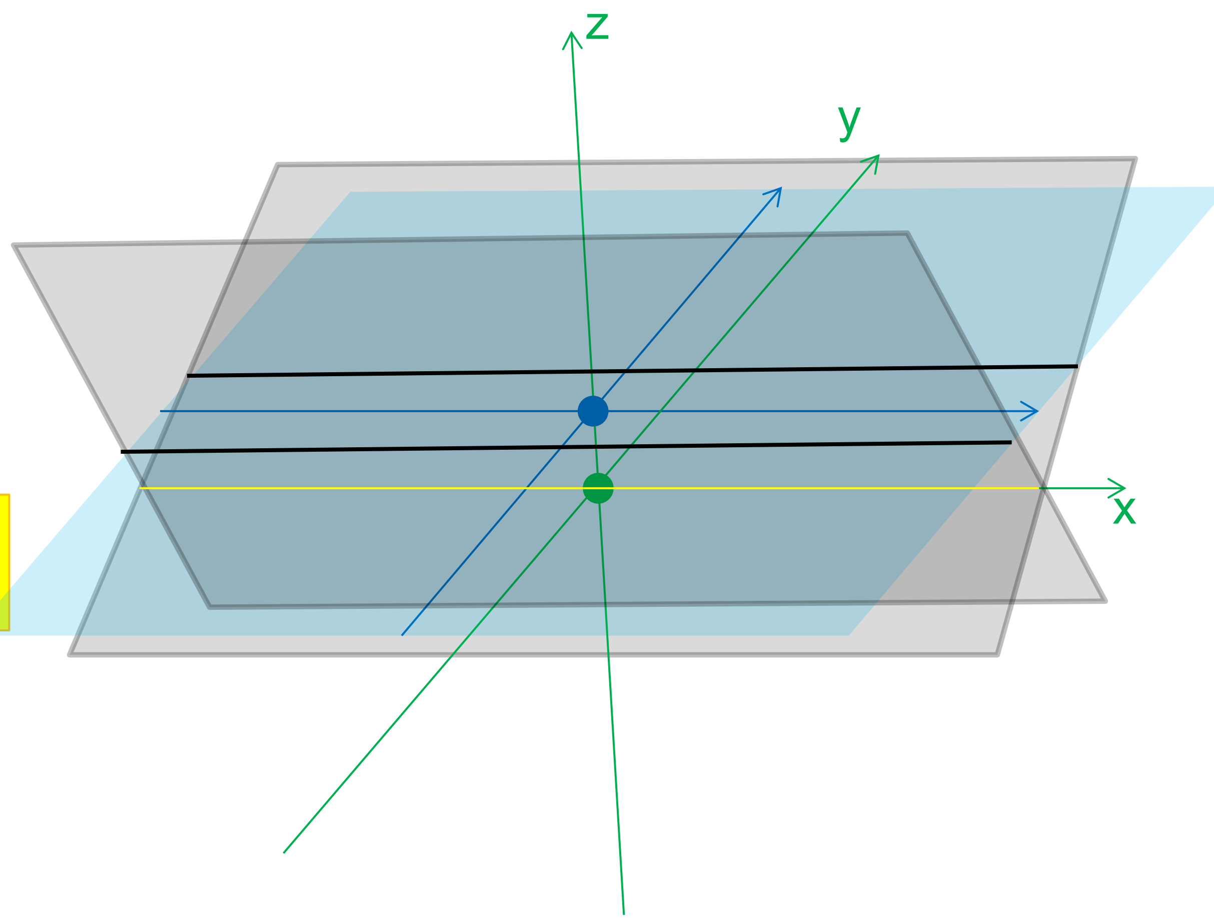
$$
\begin{tikzcd}
\text{plane-plane} \arrow[r] \arrow[d] & \text{line-line} \arrow[d] \\
\boxed{\text{line}} \arrow[r] & \boxed{\text{point}}
\end{tikzcd}
$$



Intersections

- $P^n(\mathbb{R})$ - projective space
- $\mathbb{R}^n$ - affine view

$P^n(\mathbb{R})$ | $\mathbb{R}^n$
plane-plane → parallel line-line
↓ ↓
line → point at infinity



Orthonormal Matrices

- **Columns are orthogonal vectors of unit length**
  - An example
    - $\begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$
  - Directly derived from the definition of the matrix product
    - $M^T M = 1$
  - In this case the transpose must be identical to the inverse
    - $M^{-1} \coloneqq M^T$

Linear Transformation: Matrix

- **Transformations in a Vector space: Multiplication by a Matrix**
  - Action of a linear transformation on a vector
    - Multiplication of matrix with column vector (e.g. in 3D)

$$p' = \begin{pmatrix} X' \\ Y' \\ Z' \end{pmatrix} = \boldsymbol{T}p = \begin{pmatrix} T_{xx} & T_{xy} & T_{xz} \\ T_{yx} & T_{yy} & T_{yz} \\ T_{zx} & T_{zy} & T_{zz} \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$$

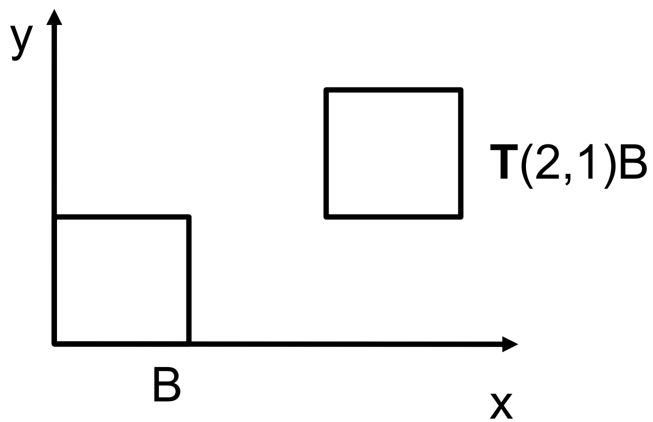
- **Composition of transformations**
  - Simple matrix multiplication ($\boldsymbol{T_1}$, then $\boldsymbol{T_2}$)
    - $\boldsymbol{T_2}\boldsymbol{T_1}p = \boldsymbol{T_2}(\boldsymbol{T_1}p) = (\boldsymbol{T_2}\boldsymbol{T_1})p = \boldsymbol{T}p$
  - Note: matrix multiplication is associative but not commutative!
    - $\boldsymbol{T_2}\boldsymbol{T_1}$ is not the same as $\boldsymbol{T_1}\boldsymbol{T_2}$ (in general)

Affine Transformation

- **Remember:**
  - Affine map: Linear mapping and a translation
    - $\boldsymbol{T}p = \boldsymbol{A}p + \boldsymbol{t}$
- **For 3D: Combining it into one matrix**
  - Using homogeneous 4D coordinates
  - Multiplication by 4x4 matrix in P(R^4) space
    - $p' = \begin{pmatrix} X' \\ Y' \\ Z' \\ W' \end{pmatrix} = Tp = \begin{pmatrix} T_{xx} & T_{xy} & T_{xz} & T_{xw} \\ T_{yx} & T_{yy} & T_{yz} & T_{yw} \\ T_{zx} & T_{zy} & T_{zz} & T_{zw} \\ T_{wx} & T_{wy} & T_{wz} & T_{ww} \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ W \end{pmatrix}$
  - Allows for combining (concatenating) multiple transforms into one using normal (4x4) matrix product
- **Let's go through the different transforms we need:**

Transformations: Translation

- **Translation (T)**
  - $\boldsymbol{T}(t_x, t_y, t_z)p = \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x + t_x \\ y + t_y \\ z + t_z \\ 1 \end{pmatrix}$



Translation of Vectors

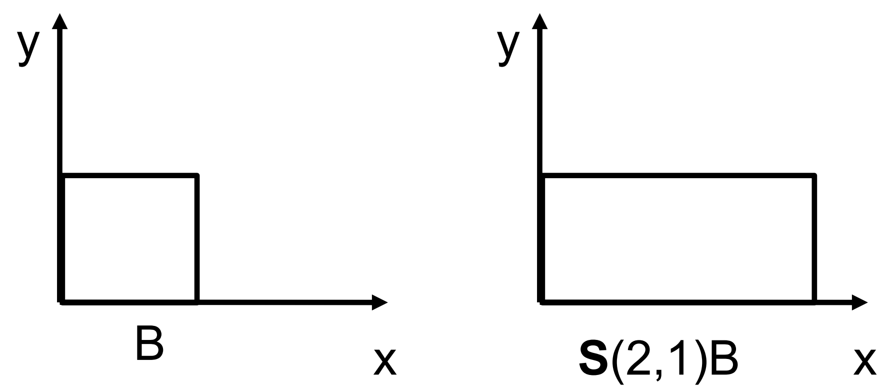
- **So far: only translated points**
- **Vectors: Difference between 2 points**
  - $v = p - q = \begin{pmatrix} p_x \\ p_y \\ p_z \\ 1 \end{pmatrix} - \begin{pmatrix} q_x \\ q_y \\ q_z \\ 1 \end{pmatrix} = \begin{pmatrix} p_x - q_x \\ p_y - q_y \\ p_z - q_z \\ 0 \end{pmatrix}$
  - Fourth component is zero
- **Consequently: Translations do not affect vectors!**
  - $\boldsymbol{T}(t_x, t_y, t_z)v = \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} v_x \\ v_y \\ v_z \\ 0 \end{pmatrix} = \begin{pmatrix} v_x \\ v_y \\ v_z \\ 0 \end{pmatrix}$

Translation: Properties

- **Properties**
  - Identity
    - $\boldsymbol{T}(0,0,0) = \mathbf{1}$ (Identity Matrix)
  - Commutative (special case)
    - $\boldsymbol{T}\left(t_x, t_y, t_z\right)\boldsymbol{T}\left(t'_x, t'_y, t'_z\right) = \boldsymbol{T}\left(t'_x, t'_y, t'_z\right)\boldsymbol{T}\left(t_x, t_y, t_z\right) = \boldsymbol{T}(t_x + t'_x, t_y + t'_y, t_z + t'_z)$
  - Inverse
    - $\boldsymbol{T}^{-1}\left(t_x, t_y, t_z\right) = \boldsymbol{T}\left(-t'_x, -t'_y, -t'_z\right)$

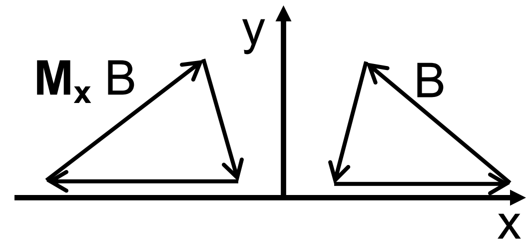
Basic Transformations (2)

- **Scaling (S)**
  - $\mathbf{S}(s_x, s_y, s_z) = \begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$
  - Note: $s_x, s_y, s_z \geq 0$ (otherwise see mirror transformation)
  - Uniform Scaling s: $s = s_x = x_y = s_z$

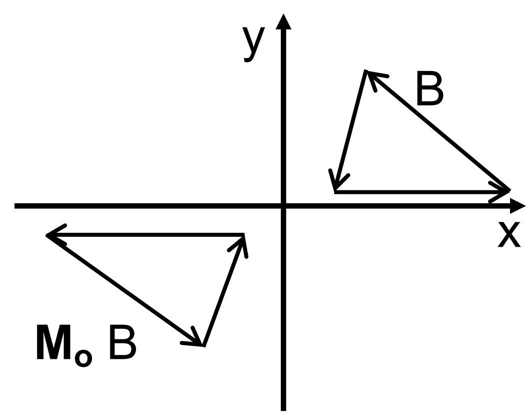


Basic Transformations

- **Reflection/Mirror Transformation (M)**
  - Reflection at plane (x=0)
    - $\boldsymbol{M_x} = \begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} -x \\ y \\ z \\ 1 \end{pmatrix}$
    - Analogously for other axis
    - Note: changes orientation
      - Right-handed becomes left-handed and v.v.
      - Indicated by $\det(M_i) < 0$



  - Reflection at origin
    - $\boldsymbol{M_o} = \begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} -x \\ -y \\ -z \\ 1 \end{pmatrix}$
    - Note: changes orientation in 3D
      - But not in 2D (!!!): Just two scale factors
      - Each scale factor reverses orientation once

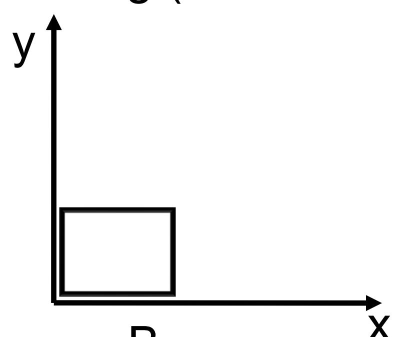


Basic Transformations (4)

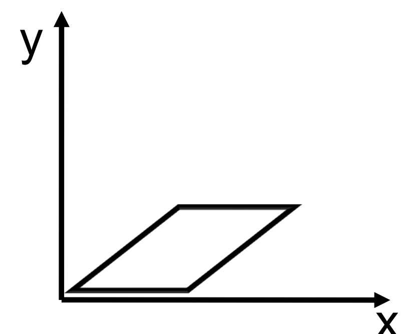
- **Shear (H)**
  - $\boldsymbol{H}\left(h_{xy}, h_{xz}, h_{yz}, h_{yx}, h_{zx}, h_{zy}\right) =$

$$\begin{pmatrix} 1 & h_{xy} & h_{xz} & 0 \\ h_{yx} & 1 & h_{yz} & 0 \\ h_{zx} & h_{zy} & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x + h_{xy}y + h_{xz}z \\ y + h_{yx}x + h_{yz}z \\ z + h_{zx}x + h_{zy}y \\ 1 \end{pmatrix}$$

  - Determinant is 1
    - Volume preserving (as volume is just shifted in some direction)



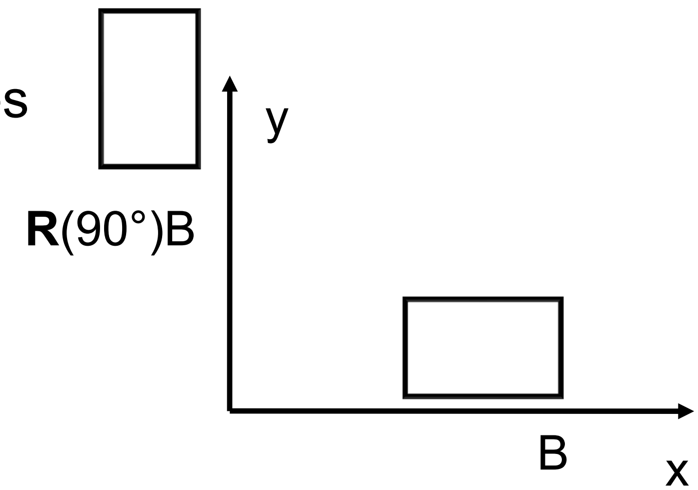
B

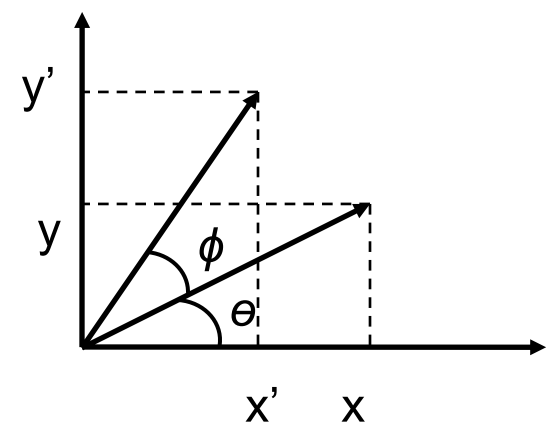


H(1,0,0,0,0,0)B

Rotation in 2D

- **In 2D: Rotation around origin**
  - Representation in spherical coordinates
    - $x = r\cos\theta \longrightarrow x' = r\cos(\theta+\phi)$
      $y = r\sin\theta \longrightarrow y' = r\sin(\theta+\phi)$
  - Well know property
    - $\cos(\theta+\phi) = \cos\theta\cos\phi - \sin\theta\sin\phi$
      $\sin(\theta+\phi) = \cos\theta\sin\phi + \sin\theta\cos\phi$
  - Gives
    - $x' = (r\cos\theta)\cos\phi - (r\sin\theta)\sin\phi = x\cos\phi - y\sin\phi$
      $y' = (r\cos\theta)\sin\phi + (r\sin\theta)\cos\phi = x\sin\phi + y\cos\phi$
  - Or in matrix form
    - $R_z(\phi) = \begin{pmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{pmatrix}$





Rotation in 3D

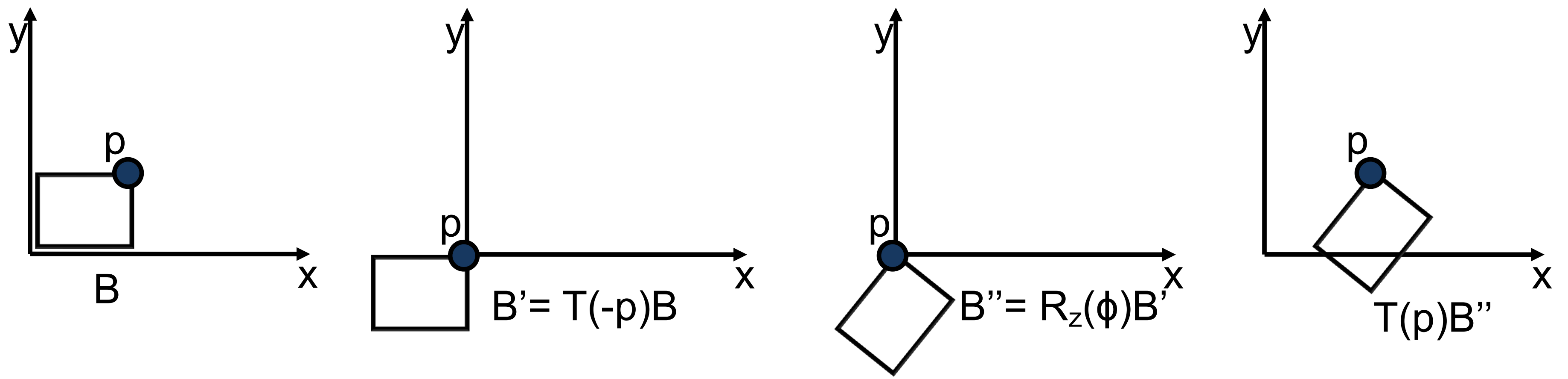
- **Rotation around major axes**
  - $\boldsymbol{R}_x(\phi) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi & 0 \\ 0 & \sin\phi & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$
  - $\boldsymbol{R}_y(\phi) = \begin{pmatrix} \cos\phi & 0 & \sin\phi & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\phi & 0 & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$
  - $\boldsymbol{R}_z(\phi) = \begin{pmatrix} \cos\phi & -\sin\phi & 0 & 0 \\ \sin\phi & \cos\phi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$
  - 2D rotation around the respective axis
    - Assumes right-handed system, mathematically positive direction
  - Be aware of change in sign on sines in $\boldsymbol{R}_y$
    - Due to relative orientation of other axis

Rotation in 3D (2)

- **Properties**
  - $\boldsymbol{R_a}(0) = \mathbf{1}$
  - $\boldsymbol{R_a}(\theta)\boldsymbol{R_a}(\phi) = \boldsymbol{R_a}(\theta + \phi) = \boldsymbol{R_a}(\phi)\boldsymbol{R_a}(\theta)$
    - Rotations around the same axis are commutative (special case)
  - In general: Not commutative
    - $\boldsymbol{R_a}(\theta)\boldsymbol{R_b}(\phi) \neq \boldsymbol{R_b}(\phi)\boldsymbol{R_a}(\theta)$
    - Order **does** matter for rotations around different axes
  - $\boldsymbol{R_a^{-1}}(\theta) = \boldsymbol{R_a}(-\theta) = \boldsymbol{R_a^T}(\theta)$
    - Orthonormal matrix: Inverse is equal to the transpose
  - Determinant is 1
    - Volume preserving

Rotation Around Point

- **Rotate object around a point p and axis a**
  - Translate p to origin, rotate around axis a, translate back to p
    - $\boldsymbol{R_a}(p,\theta) = \boldsymbol{T}(p)\boldsymbol{R_a}(\phi)\boldsymbol{T}(-p)$



Rotation Around Some Axis

- Rotate around a given point p and vector r (|r|=1)
  - Translate so that p is in the origin
  - Transform with rotation $R=M^T$
    - M given by orthonormal basis (r,s,t) such that r becomes the **x** axis
    - Requires construction of a orthonormal basis (r,s,t), see next slide
  - Rotate around **x** axis
  - Transform back with R^{-1}
  - Translate back to point p

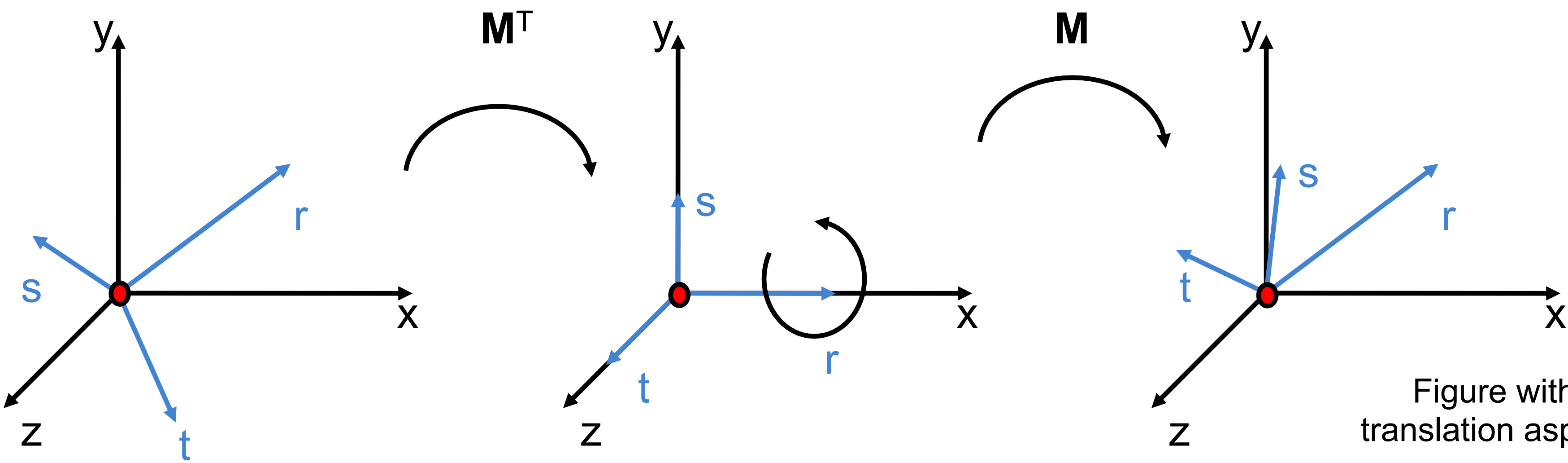


Figure without translation aspect

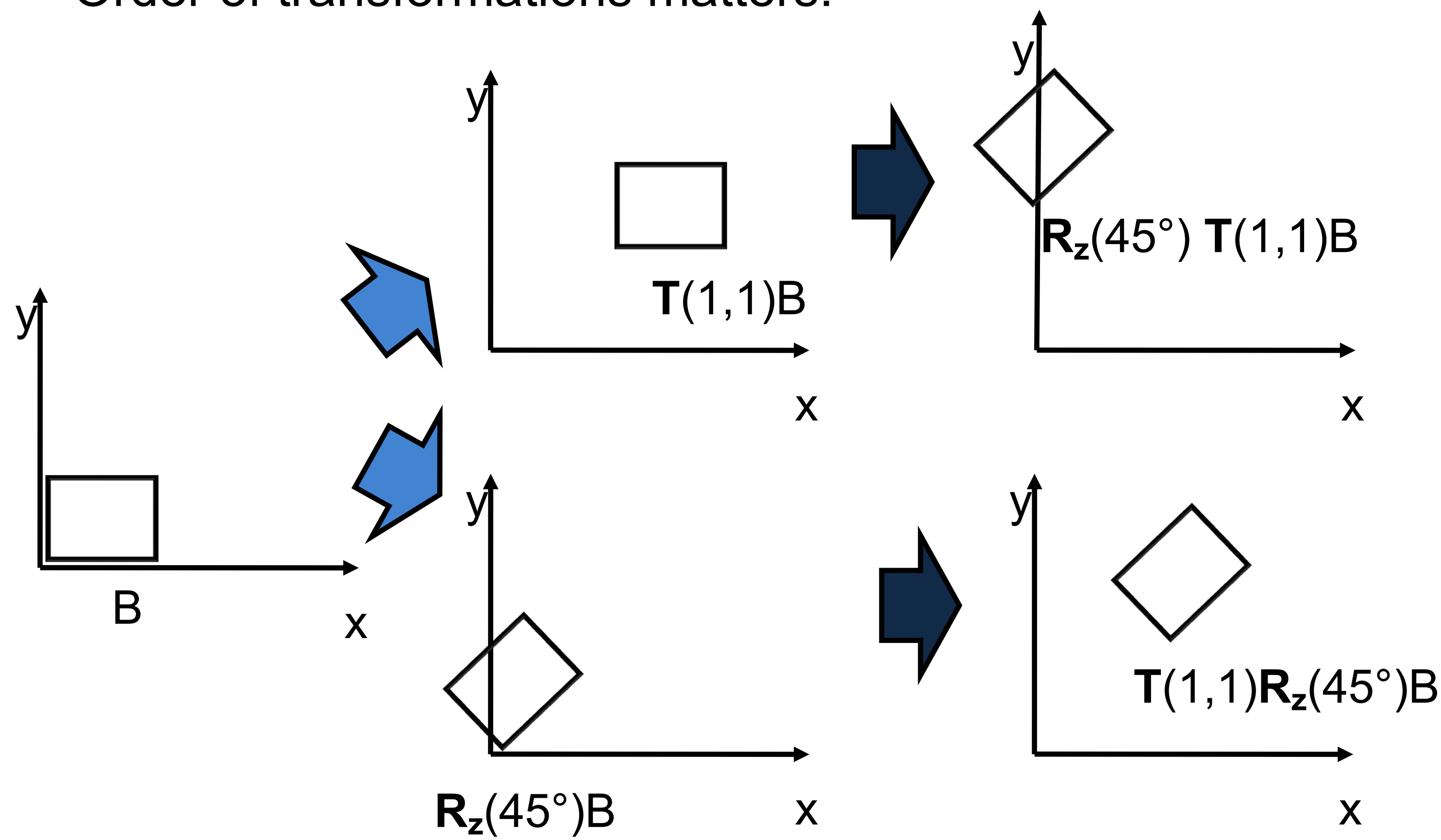
$$R(p,r,\phi) = T(p)M(r)R_x(\phi)M^T(r)\mathrm{T}(-\mathrm{p})$$

Rotation Around Some Axis

- **Compute orthonormal basis given a vector r**
  - Using a numerically stable method
  - Construct s such that its normal to r (verify with dot product)
    - Use fact that in 2D, orthogonal vector to (x,y) is (-y, x)
      - Do this in coordinate plane that has largest components
    - $s' = \begin{cases} (0, -r_z, r_y), \text{if } x = \text{argmin}_{x,y,z}\{|r_x|, |r_y|, |r_z|\} \\ (-r_z, 0, r_x), \text{if } y = \text{argmin}_{x,y,z}\{|r_x|, |r_y|, |r_z|\} \\ (-r_y, r_x, 0), \text{if } z = \text{argmin}_{x,y,z}\{|r_x|, |r_y|, |r_z|\} \end{cases}$
  - Normalize
    - $s = s'/|s'|$
  - Compute t as cross product
    - $t = r \times s$
  - r,s,t forms orthonormal basis, thus M transforms into this basis
    - $M(r) = \begin{pmatrix} r_x & s_x & t_x & 0 \\ r_y & s_y & t_y & 0 \\ r_z & s_z & t_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$, inverse is given as its transpose: $M^{-1} = M^T$

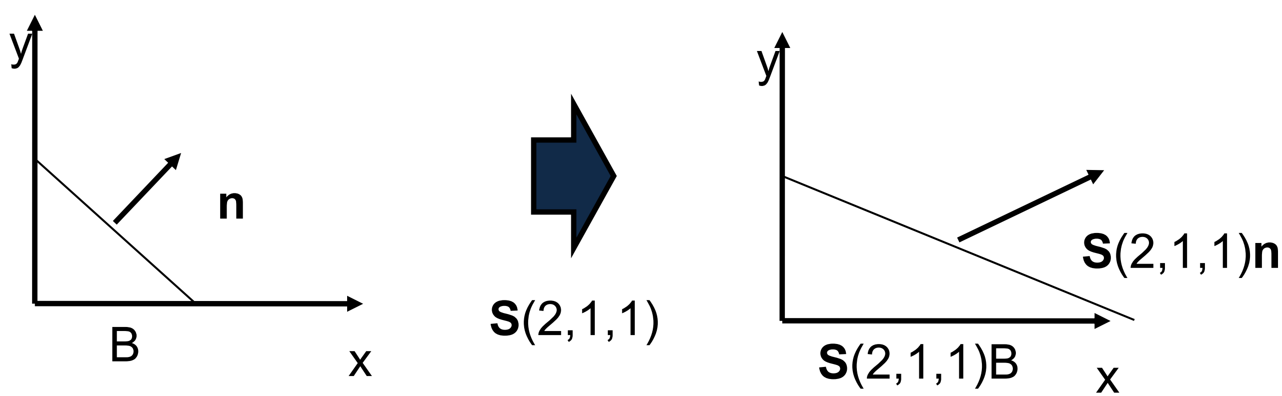
Concatenation of Transforms

- **Multiply matrices to concatenate**
  - Matrix-matrix multiplication is not commutative (in general)
  - Order of transformations matters!



Transformations

- **Line**
  - Transform end points
- **Plane**
  - Transform three points
- **Vector**
  - Translations to not act on vectors
- **Normal vectors**
  - Problem: e.g. with non-uniform scaling



Transforming Normals

- **Dot product as matrix multiplication**
  - $n \cdot v = n^T v = \begin{pmatrix} n_x & n_y & n_z \end{pmatrix} \begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix}$
- **Normal N on a plane**
  - For any vector v in the plane: $n^T v = 0$
  - Find transformation ***M'*** for normal vector, such that :
    - $(\boldsymbol{M}' n)^T (\boldsymbol{M} v) = 0$
      $n^T (\boldsymbol{M}'^T \boldsymbol{M}) v = 0$ and thus
      $\boldsymbol{M}'^T \boldsymbol{M} = 1$

      $\boldsymbol{M}'^T \boldsymbol{M} \boldsymbol{M}^{-1} = 1 \boldsymbol{M}^{-1}$
      $\boldsymbol{M}'^T = \boldsymbol{M}^{-1}$
      $\boldsymbol{M}' = \boldsymbol{M}^{-1T}$
  - ***M'*** is the adjoint of ***M***
    - Exists even for non-invertible matrices
    - For ***M*** invertible and orthogonal $M' = (M^{-1})^T = (M^T)^T = M$
- **Remember:**
  - Normals are transformed by the transpose of the inverse of the 4x4 transformation matrix of points and vectors

USING TRANSFORMATIONS

Coordinate Systems

- **Local (object) coordinate system (3D)**
  - Object vertex positions
  - Can be hierarchically nested in each other (scene graph, transf. stack)
- **World (global) coordinate system (3D)**
  - Scene composition and object placement
    - Rigid objects: constant translation, rotation per object, (scaling)
    - Animated objects: time-varying transformation in world-space
  - Illumination can be computed in this space

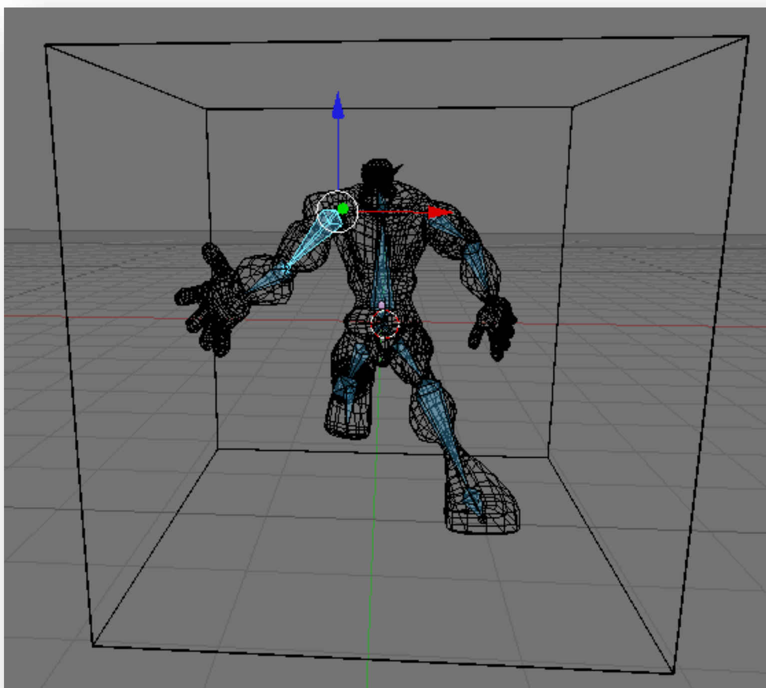
Hierarchical Coordinate Systems

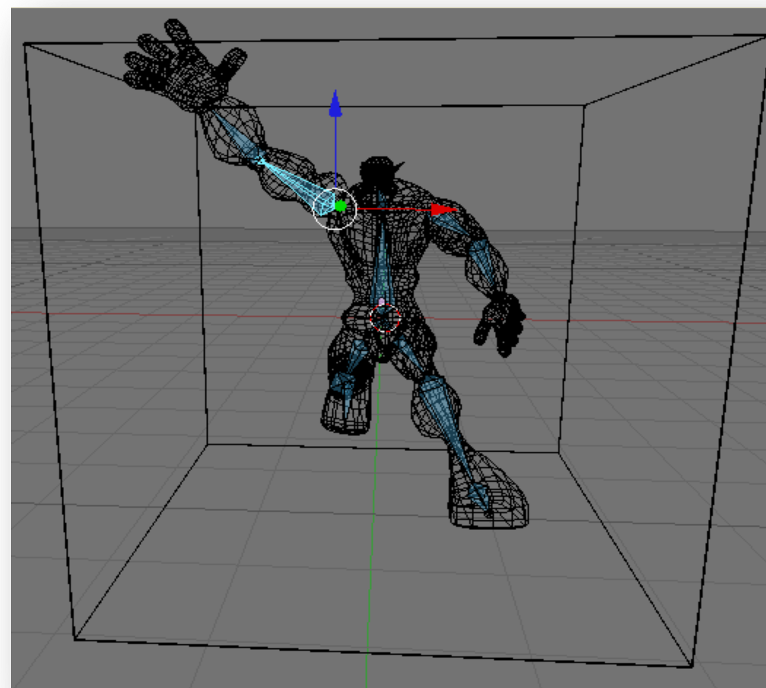
- **Hierarchy of transformations**

```
T_root                      //Position of the character in world
  T_ShoulderR               //Right shoulder position
    T_ShoulderRJoint        //Shoulder rotation  <== User
      T_UpperArmR           //Elbow position
        T_ElbowRJoint       //Elbow rotation     <== User
          T_LowerArmR       //Wrist position
            T_WristRJoint   //Wrist rotation     <== User
              ...           //Hand and fingers...
  T_ShoulderL               //Left shoulder position
    T_ShoulderLJoint        //Shoulder rotation  <== User
      T_UpperArmL           //Elbow position
        T_ElbowLJoint       //Elbow rotation     <== User
          T_LowerArmL       //Wrist position
            ...
```

Hierarchical Coordinate Systems

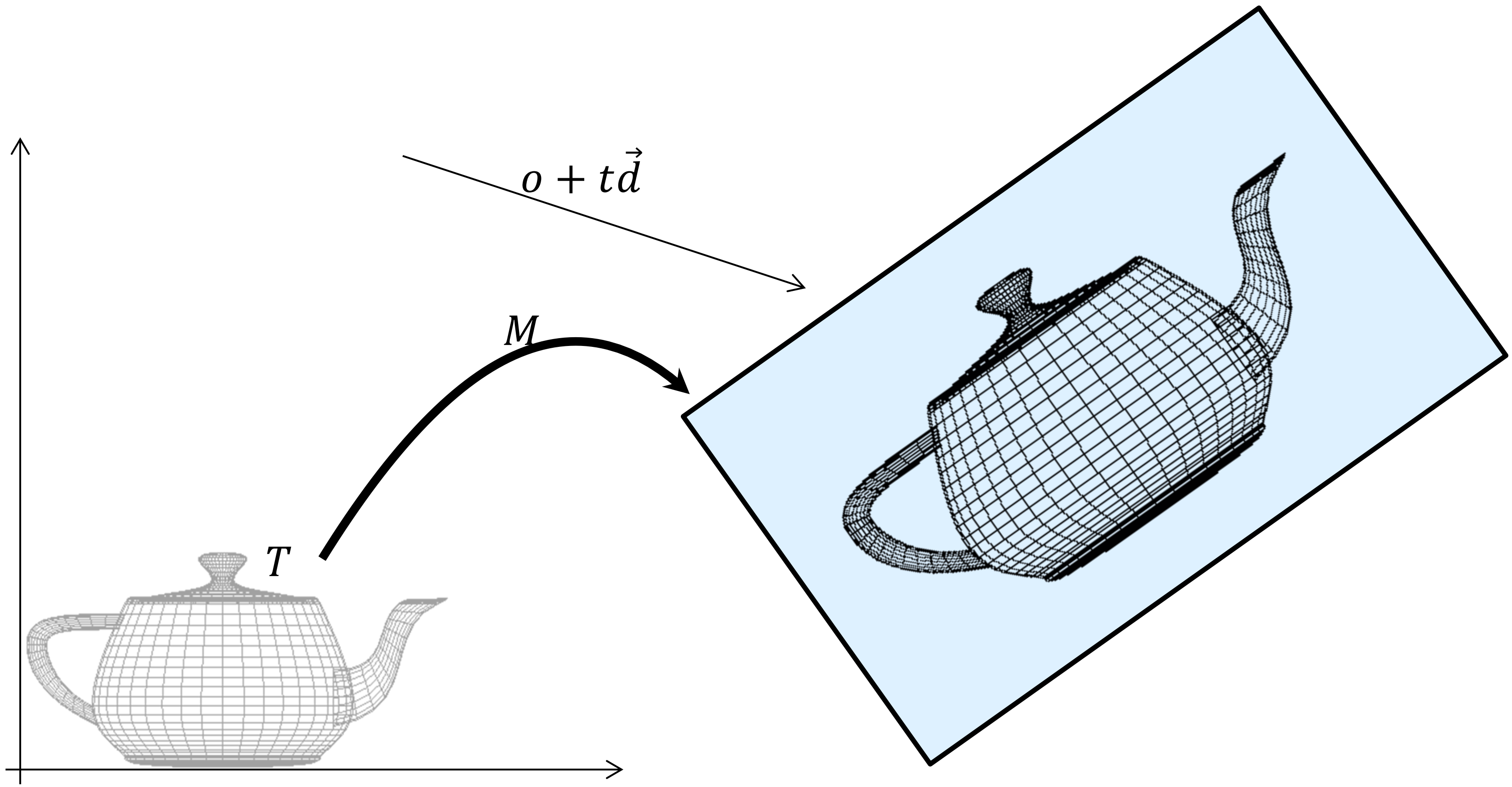
- **Used in Scene Graphs**
  - Group objects hierarchically
  - Local coordinate system is relative to parent coordinate system
  - Apply transformation to the parent to change the whole sub-tree (or sub-graph)





Ray-tracing Transformed Objects

- Ray (world coordinates)
- T – set of triangles (local coordinates)
- M – transformation matrix (local-to-world)

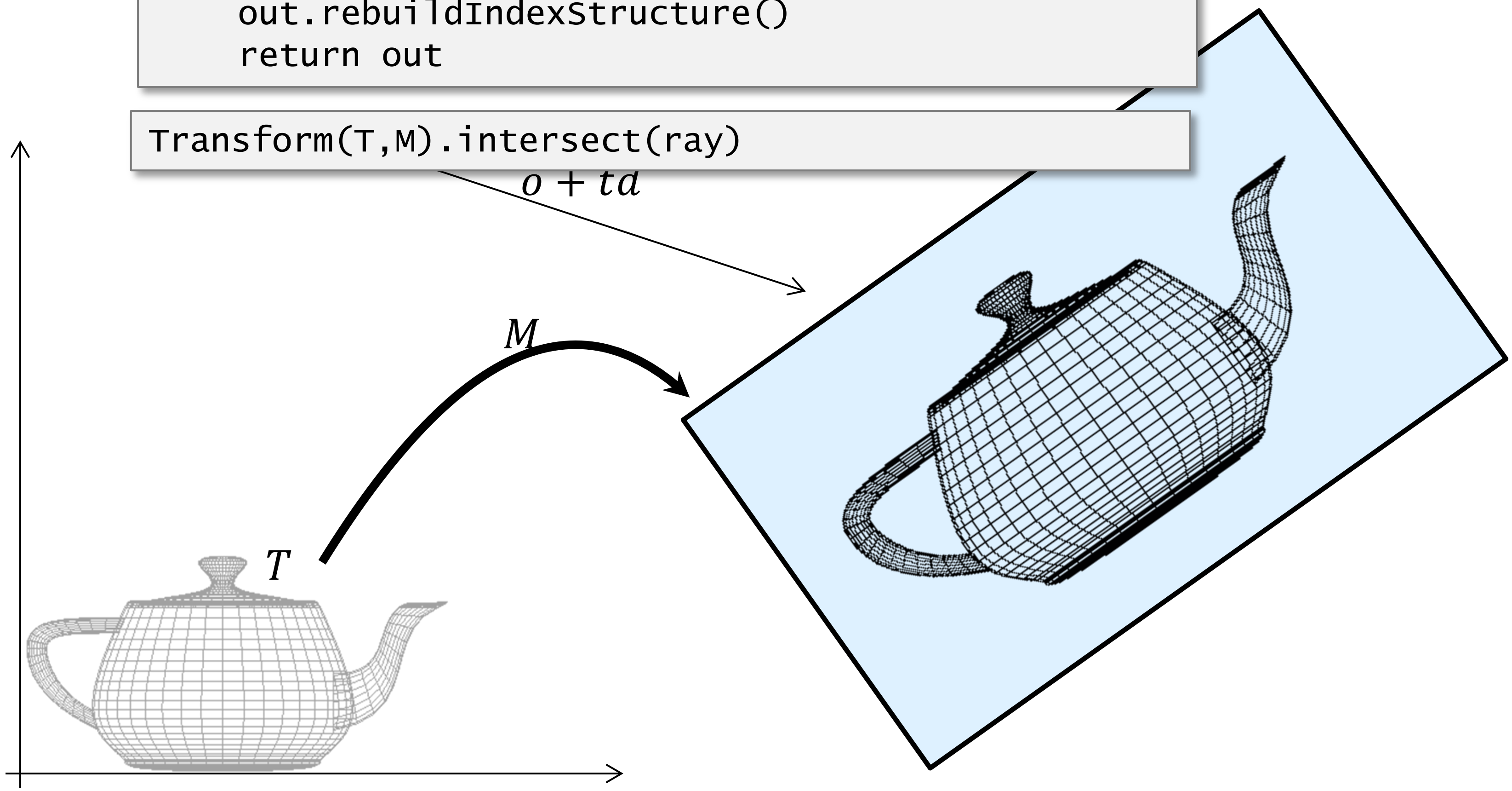


Ray-tracing Transformed Objects

- Option 1: transform the triangles

```
def transform(T,M)
    out = {}
    foreach p in T
        q = M*p
        out.insert(q)
    out.rebuildIndexStructure()
    return out
```

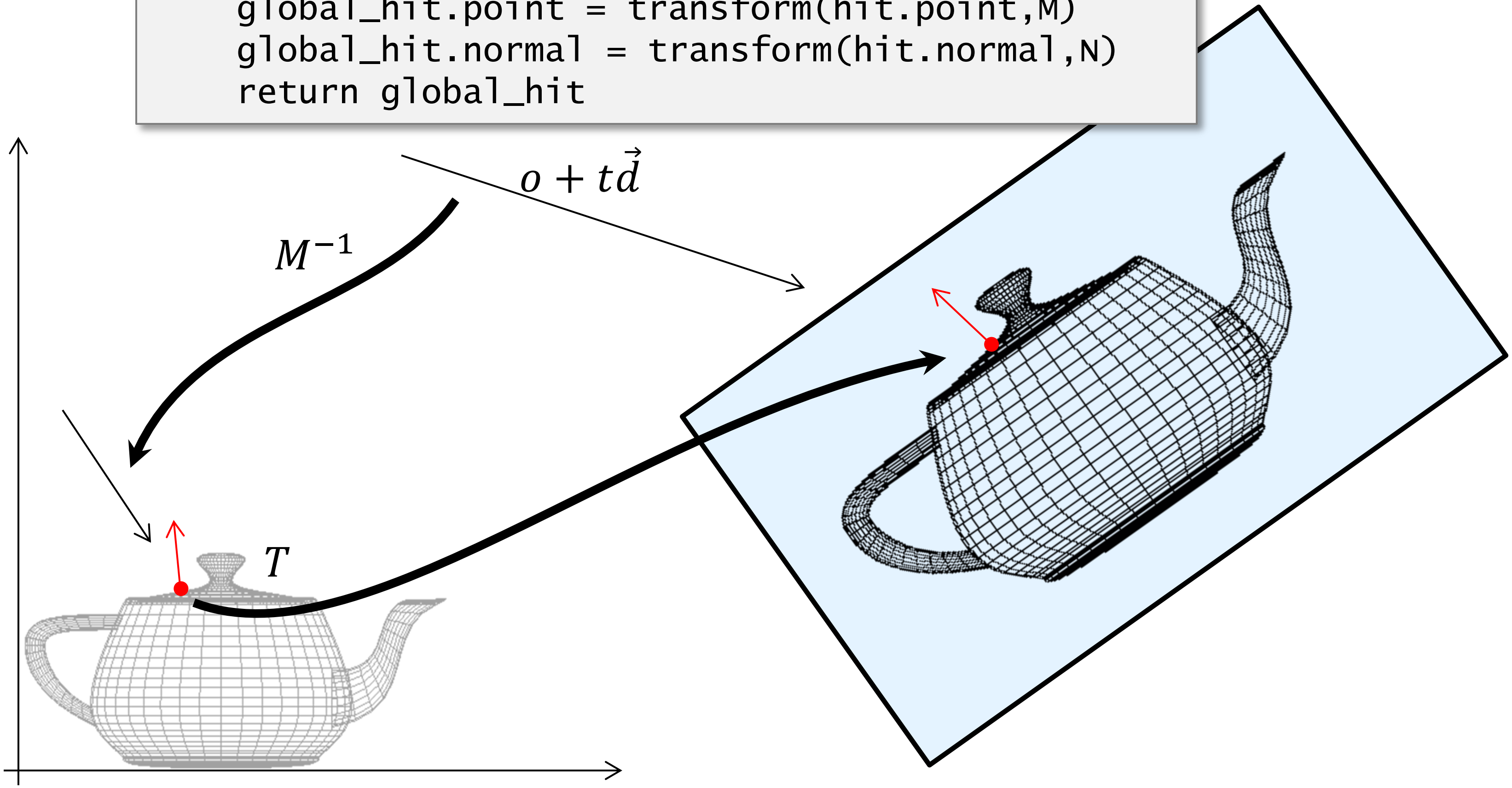
```
Transform(T,M).intersect(ray)
```



Ray-tracing Transformed Objects

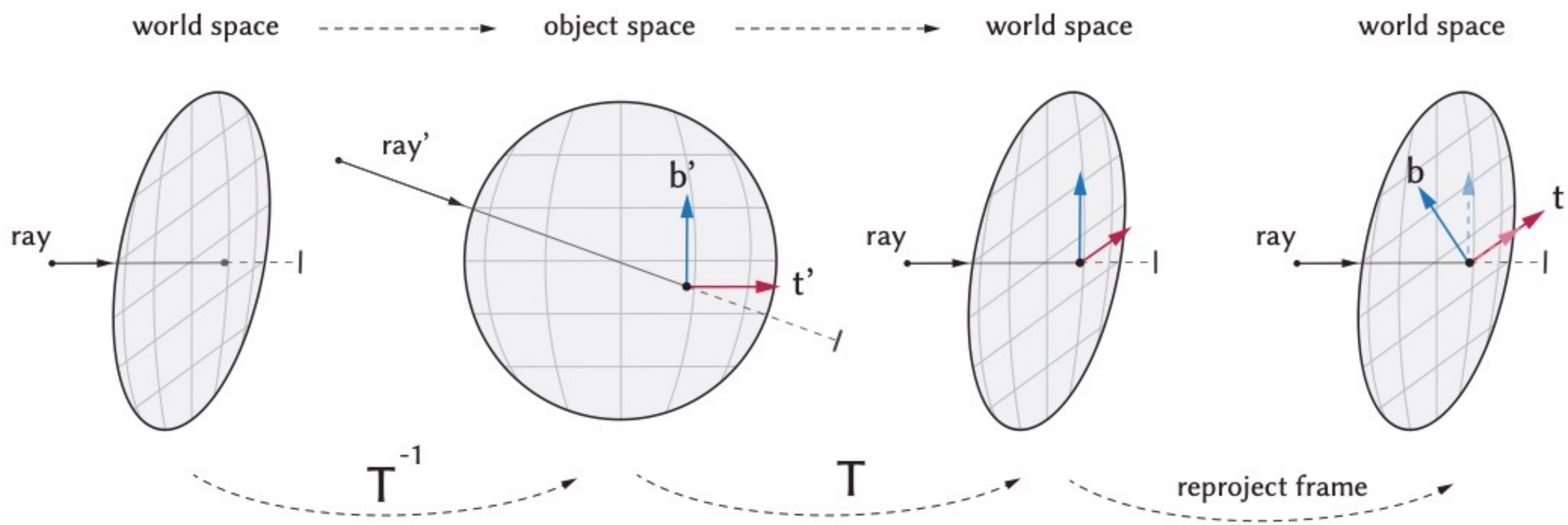
- Option 2: transform the ray

```
def intersect(obj,ray)
    Minv = obj.M.inverse()
    N = obj.M.normalTransform()
    local_ray = transform(ray,Minv)
    hit = obj.intersect(local_ray)
    global_hit.point = transform(hit.point,M)
    global_hit.normal = transform(hit.normal,N)
    return global_hit
```



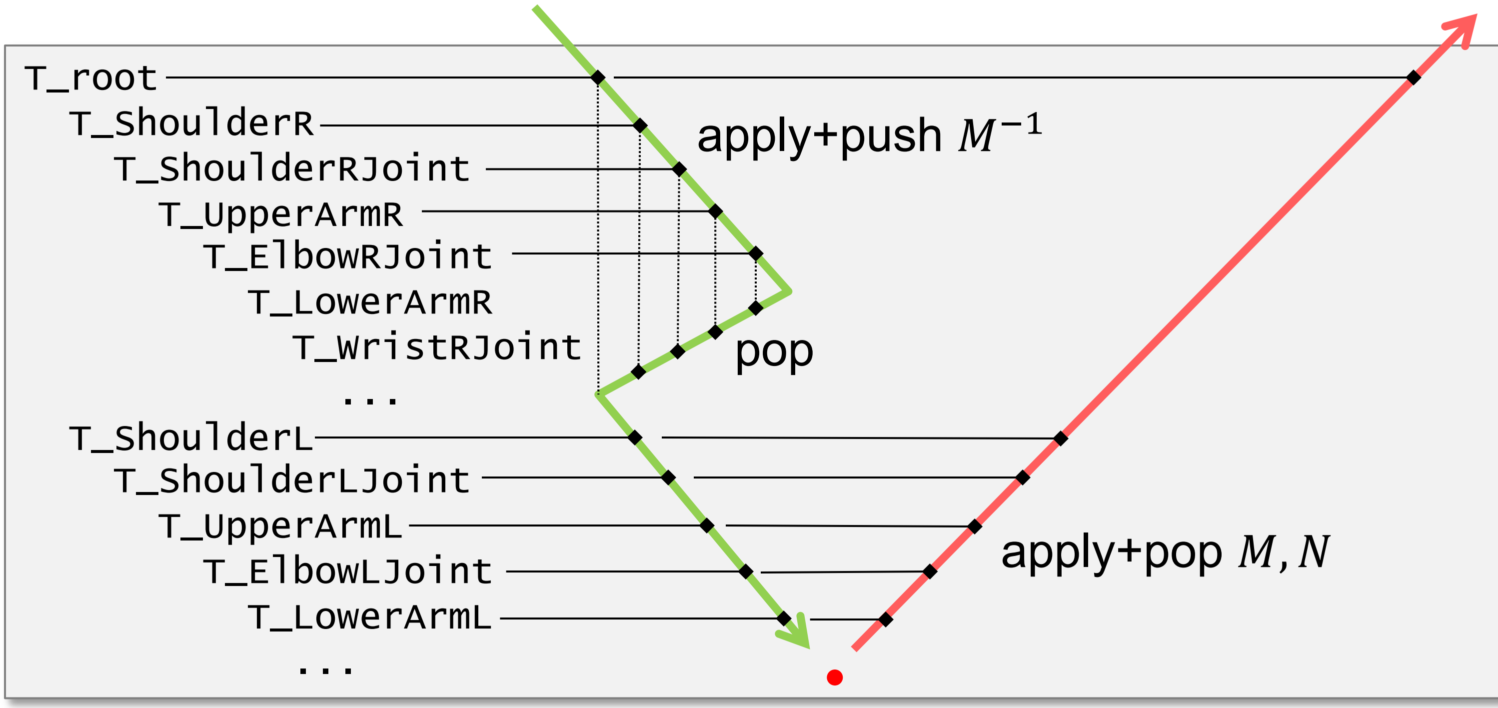
Transforming Tangents

- **Transform ray by inverse and intersect object…**



- **…then transform tangents back to world space**
  – Bitangent might need to be adjusted to obtain orthonormal basis
  – Adjoint matrix not necessary, can compute normal from tangent and bitangent

Ray-tracing through a Hierarchy



Instancing

- T – set of triangles
  - local coordinates
  - memory
- M_i – transformation matrices
  - local-to-world
- Multiple rendered objects
  - Correct lighting, shadows, etc...
  - Never ”materialized” in memory