Computer Graphics

- Transformations -

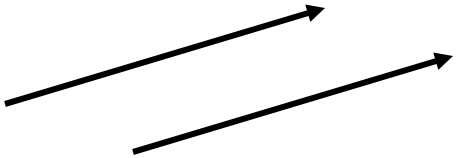
Philipp Slusallek

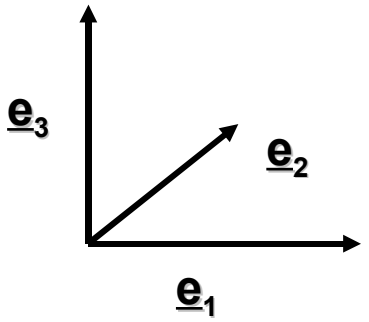
Overview

- **Last Time**
  - Color Spaces
- **Today**
  - Homogeneous Coordinates
  - Basic transformations in homogeneous coordinates
  - Concatenation of transformations
  - Projective transformations
- **Next Lecture**
  - Spline Curves

Vector Space

- **Known from Mathematics:**
  - Elements of a 3D vector space
    - $\underline{v} = (v_1, v_2, v_3)^T \in V^3 = R^3$
  - Formally: Vectors written as column vectors (n x 1 matrix)!
  - Vectors describe directions – not positions!
  - 3 linear independent vectors create a basis:
    - $\{\underline{e}_1, \underline{e}_2, \underline{e}_3\}$
  - Any vector can now uniquely be represented with coordinates
    - $\underline{v} = v_1 \underline{e}_1 + v_2 \underline{e}_2 + v_3 \underline{e}_3 = (v_1, v_2, v_3)^T$
  - Operations
    - Addition, Subtraction, Scaling, …

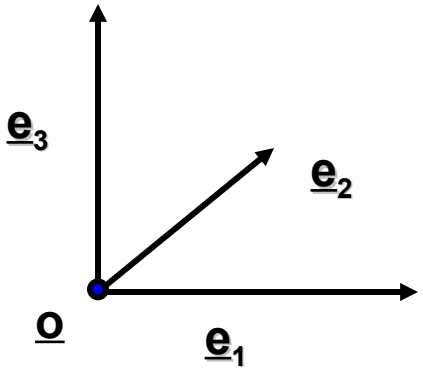




- **Metric**
  - Dot/inner product:
    - Used for measurements of length ($|\underline{v}|^2 = \underline{v} \cdot \underline{v}$) and angles ($\cos(v_1, v_2) = v_1 \cdot v_2 / |v_1||v_2|$)
  - Orthonormal basis
    - $|\underline{e}_i| = 1$ $\quad \underline{e}_i \cdot \underline{e}_j = \delta_{ij}$ $\quad$ right-/left handed: $\underline{e}_1 \times \underline{e}_2 = +/- \underline{e}_3$

Euclidian Affine Space

- **Known from Mathematics**
  - Affine Space: A^3
    - Elements are positions – no directions!
  - Defined via its associated vector space V^3
    - $a, b \in A^3 \Leftrightarrow v \in V^3$ with $v = b - a$
      - $\rightarrow$: unique, $\leftarrow$: ambiguous
      - Addition of points and vectors ($p + v \in A^3$)
    - distance(a, b) = length(a - b)
  - Operations on A^3
    - Subtraction giving a vector, but no addition
- **Affine Basis:**
  - {o, e1, e2, e3}
    - Origin: $o \in A^3$ und
    - Basis of vector space
  - Position vector of point p
    - $(p - o) \in V3$



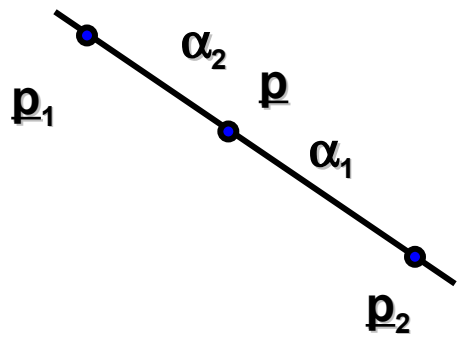
Affine Coordinates

- **Affine Combination**
  - Linear combination of (n+1) points
  - Weights form a partition of unity
  - $b_0, \ldots, b_n \in A^n$

$$\underline{b} = \sum_{i=0}^{n} \alpha_i \underline{b}_i = \underline{b}_0 + \sum_{i=1}^{n} \alpha_i (\underline{b}_i - \underline{b}_0) = \underline{o} + \sum_{i=1}^{n} \alpha_i \underline{e}_i, \quad \text{mit} \sum \alpha_i = 1$$

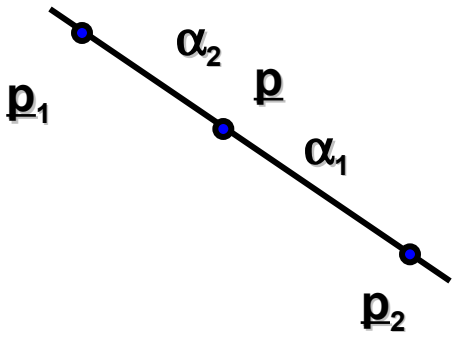
- **Affine Coordinates**
  - Barycentric coordinates
  - Center of mass ($R = \Sigma m_i r_i / \Sigma m_i$)
  - Affine weighted sum
    - Weights given by the splitting ratio
      - $\underline{p} = \alpha_1 \underline{p}_1 + \alpha_2 \underline{p}_2$

      $\alpha_1 + \alpha_2 = 1$



Affine Mappings

- **Properties**
  - Affine mapping (continuous, bijective, invertible)
    - $T: A^3 \rightarrow A^3$
  - Defined by two non-degenerated simplicies
    - 2D: Triangle, 3D: Tetrahedron, ...
  - Affine/Barycentric coordinates are invariant under affine transformations
  - Other invariants
    - Straight lines, parallelism, splitting ratios, surface/volume ratios
  - Characterization via fixed points and lines
    - Eigenvalues and eigenvectors of the mapping
- **Representation**
  - Linear mapping A plus a translation t
    - $\mathbf{T}\underline{p} = \mathbf{A}\,\underline{p} + \underline{t}$ with (n x n) matrix **A**
  - Invariance of affine coordinates
    - $Tp = T(\alpha_1\underline{p}_1 + \alpha_2\underline{p}_2) = A(\alpha_1\underline{p}_1 + \alpha_2\underline{p}_2) + \underline{t} =$ $\alpha_1 A(\underline{p}_1) + \alpha_2 A(\underline{p}_2) + \alpha_1\underline{t} + \alpha_2\underline{t} = \alpha_1 T\underline{p}_1 + \alpha_2\, T(\underline{p}_2)$



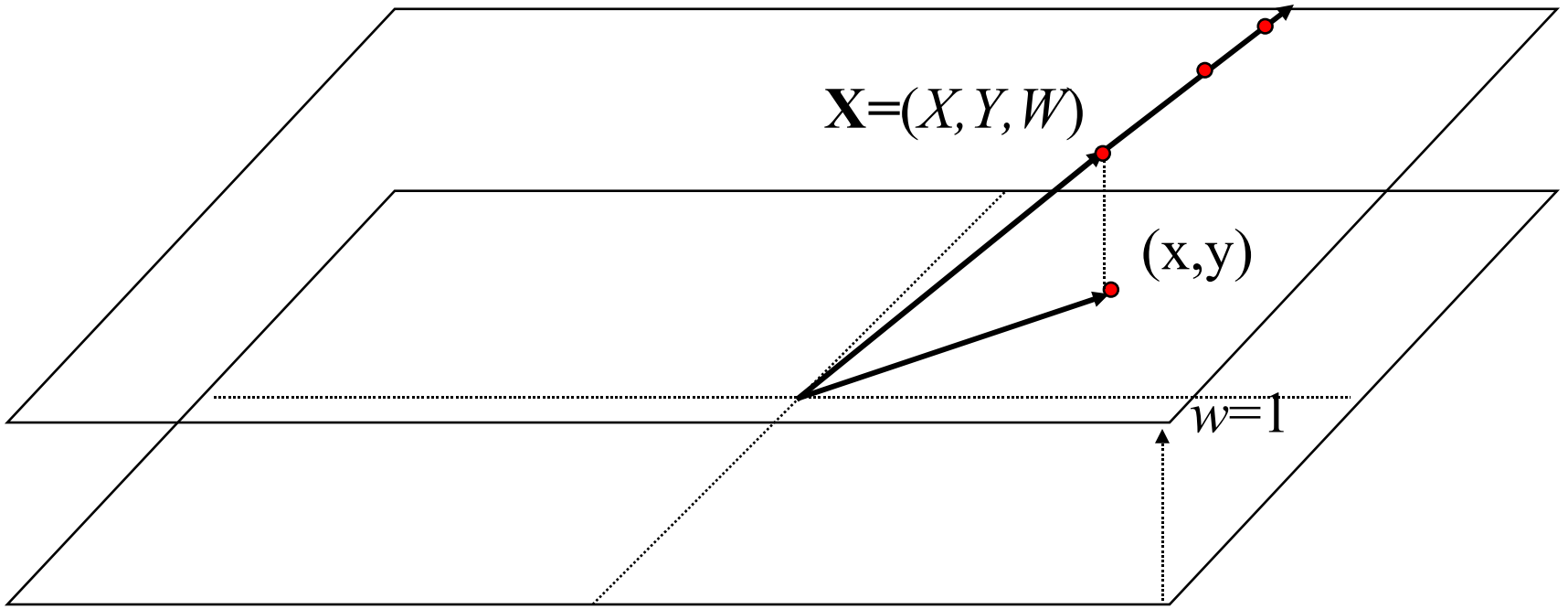
Homogeneous Coordinates for 3D

- **Embedding of R^3 into $P(R^4)$**
  - For the time being

$$R^3 \ni \begin{pmatrix} x \\ y \\ z \end{pmatrix} \rightarrow \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \in P(R^4), \quad \text{and} \begin{pmatrix} X \\ Y \\ Z \\ W \end{pmatrix} \rightarrow \begin{pmatrix} X/W \\ Y/W \\ Z/W \end{pmatrix}$$

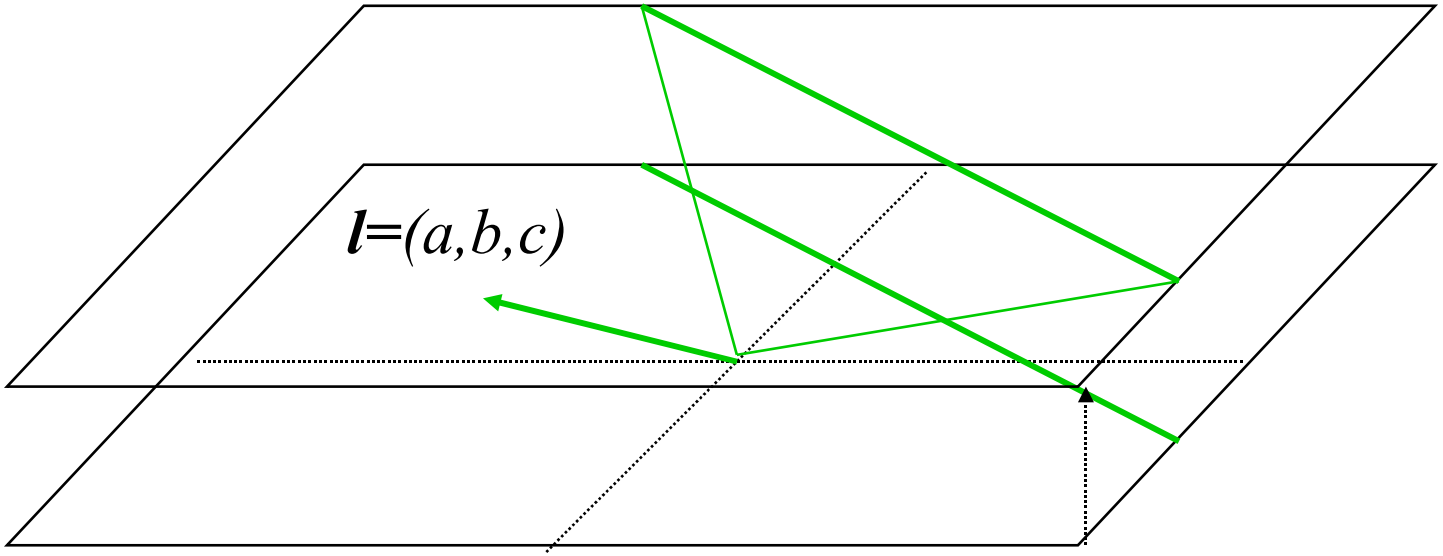
  - Representation of transformations by 4x4 matrices
  - Mathematical trick
    - Convenient representation to express rotations and translations as matrix multiplications
    - Easy to find line through points, point-line/line-line intersections
  - Also important for projections (later)

Point Representation



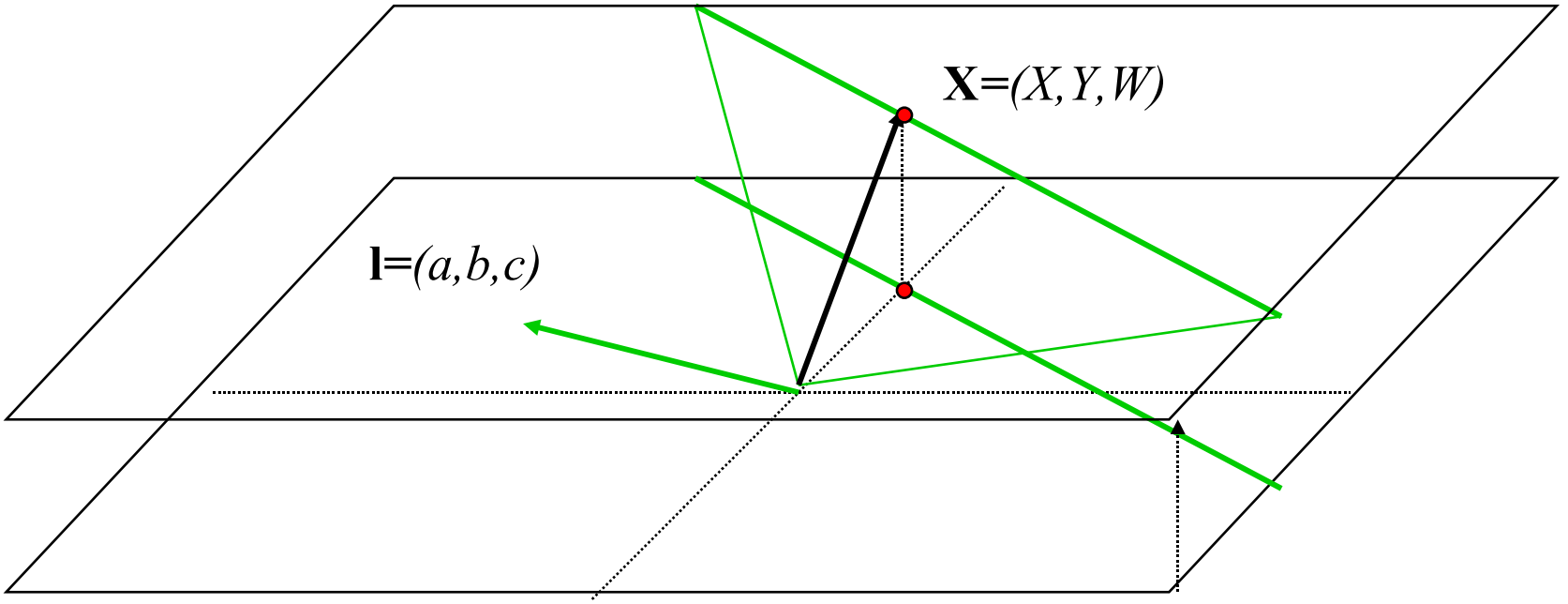
$$x = \frac{X}{W} \qquad y = \frac{Y}{W}$$

Line Representation



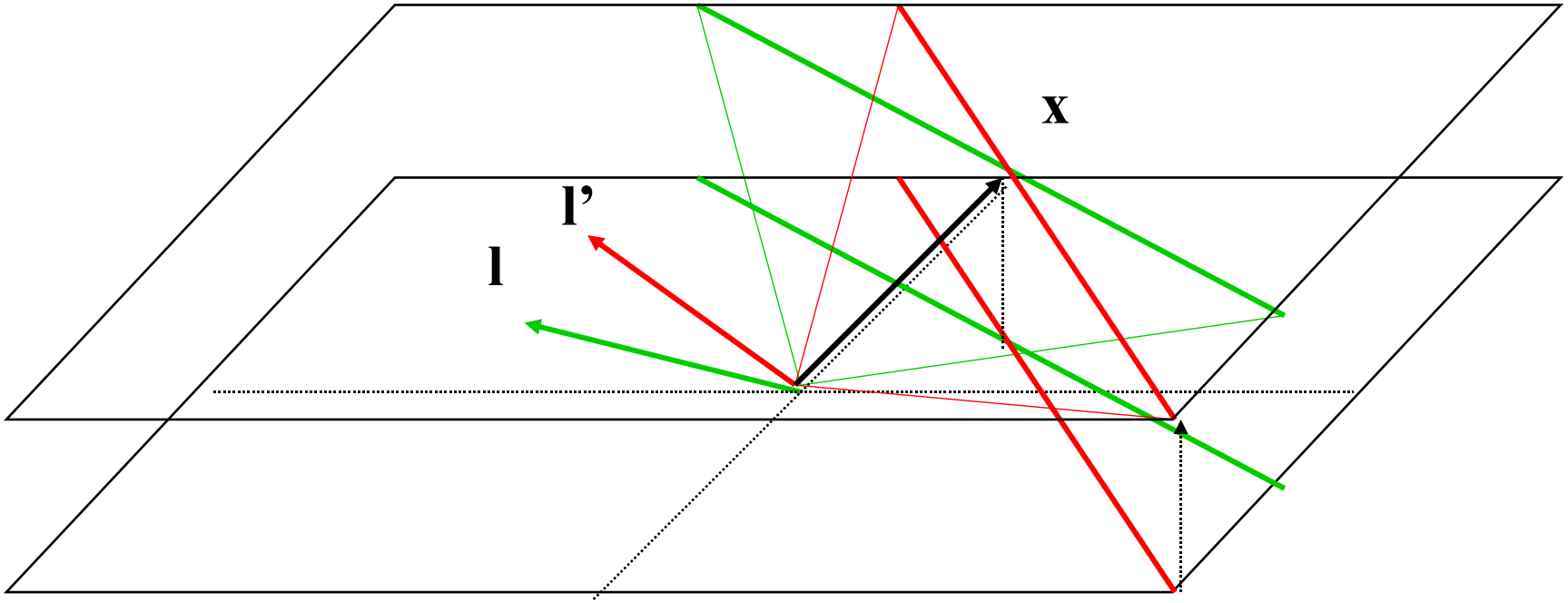
$$ax+by+c=0$$

Point on Line



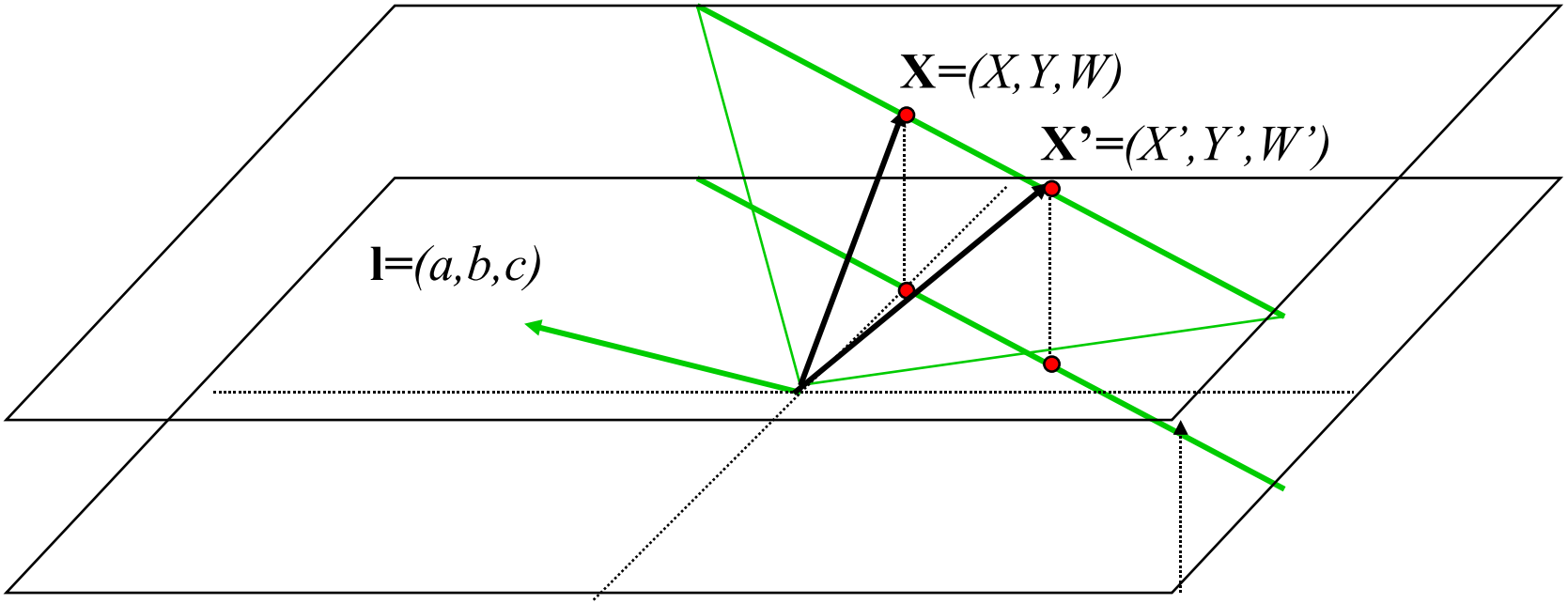
$$\mathbf{x} \cdot \mathbf{l} = 0$$

Intersection of Lines



$$\mathbf{l}' \times \mathbf{l} = \mathbf{x}$$

Line through 2 Points



$$\mathbf{x}' \perp \mathbf{x} = \mathbf{l}$$

Linear Map = Matrix

- **Vector Matrix Product**
  - Action of a linear map on a vector
    - Multiplication of matrix with column vector

$$\underline{p}' = \begin{pmatrix} x' \\ y' \\ z' \\ w' \end{pmatrix} = T\,\underline{p} = T \begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix} = \begin{pmatrix} t_{xx} & t_{xy} & t_{xz} & t_{xw} \\ t_{yx} & t_{yy} & t_{yz} & t_{yw} \\ t_{zx} & t_{zy} & t_{zz} & t_{zw} \\ t_{wx} & t_{wy} & t_{wz} & t_{ww} \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix}$$

    - In some old text books: row vector times the transpose of this matrix
  - Composition (first T_1, then T_2)
    - Matrix multiplication
    - $T_2 T_1 \underline{p} = T_2(T_1 \underline{p}) = (T_2 T_1)\,\underline{p} = T\,\underline{p}$
    - Warning: In general, matrix multiplication does not commute !!!

Basic Transformations

- **Translation**

$$T(d_x, d_y, d_z) = \begin{pmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\mathbf{T}\,\underline{p} = \begin{pmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} p_x \\ p_y \\ p_z \\ 1 \end{pmatrix} = \begin{pmatrix} p_x + d_x \\ p_y + d_y \\ p_z + d_z \\ 1 \end{pmatrix}$$

Translation of Vectors

- **So far we looked at points (affine entities)**
- **Vectors are defined as the difference of two points**
- **Consequently, for vectors W is always equal to zero**

$$\underline{v} = \underline{p} - \underline{q} = \begin{pmatrix} p_x \\ p_y \\ p_z \\ 1 \end{pmatrix} - \begin{pmatrix} q_x \\ q_y \\ q_z \\ 1 \end{pmatrix} = \begin{pmatrix} p_x - q_x \\ p_y - q_y \\ p_z - q_z \\ 0 \end{pmatrix}$$

- **This means that translations *DO NOT* act on vectors**
  - Which is exactly what we expect to happen

$$\mathbf{T}\underline{v} = \begin{pmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} v_x \\ v_y \\ v_z \\ 0 \end{pmatrix} = \begin{pmatrix} v_x \\ v_y \\ v_z \\ 0 \end{pmatrix} = \underline{v}$$

Translations

- **Properties**
  - $T(0,0,0) = \mathbf{1}$ (Identity Matrix)
  - $T(t_x, t_y, t_z)\, T(t_x', t_y', t_z') = T(t_x + t_x', t_y + t_y', t_z + t_z')$
  - $T(t_x, t_y, t_z)\, T(t_x', t_y', t_z') = T(t_x', t_y', t_z')\, T(t_x, t_y, t_z)$
  - $T^{-1}(t_x, t_y, t_z) = T(-t_x, -t_y, -t_z)$

Basic Transformations

- **Rotation around major axis**

$$R_x(\theta) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad R_y(\theta) = \begin{pmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$R_z(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

  - Assumes a right handed coordinate system

Rotation

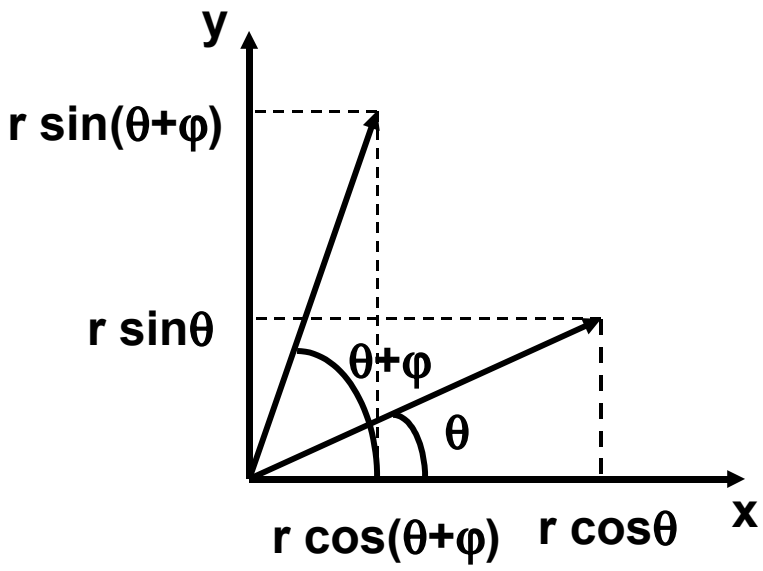
- **Rotation in 2D**

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

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

$$\cos(\theta + \varphi) = \cos\theta\cos\varphi - \sin\theta\sin\varphi$$
$$\sin(\theta + \varphi) = \cos\theta\sin\varphi + \sin\theta\cos\varphi$$

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



Rotation

- **Properties**
  - $R_a(0) = \mathbf{1}$
  - $R^{-1}{}_a(\theta) = R_a(-\theta)$
  - $R_a(\theta)\, R_a(\varphi) = R_a(\theta + \varphi)$
  - $R_a(\theta)\, R_a(\varphi) = R_a(\varphi)\, R_a(\theta)$
  - $R^{-1}{}_a(\theta) = R_a(-\theta) = R^{T}{}_a(\theta)$
  - BUT in general: $R_a(\theta)\, R_b(\varphi) \neq R_b(\varphi)\, R_a(\theta)$
    - For rotations around different axes, the order matters

Basic Transformations

- **Scaling**

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

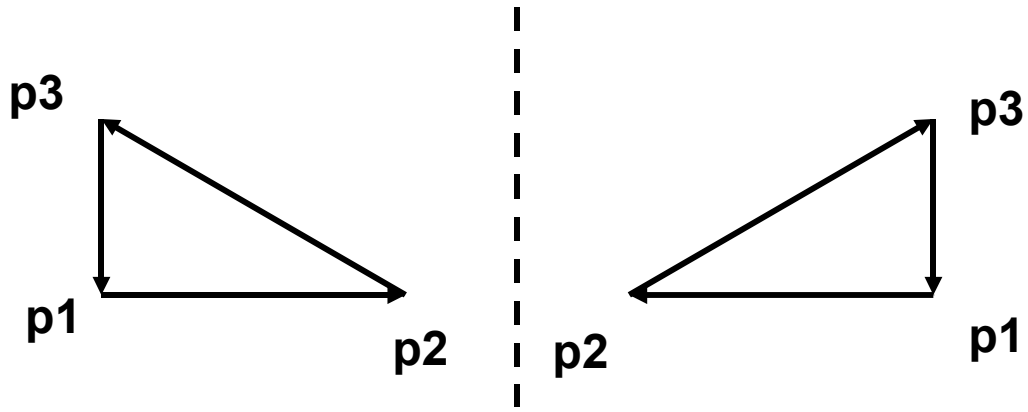
  - Uniform Scaling
    - $s_x = s_y = s_z$

Basic Transformations

- **Reflection at Z**

$$M_z = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

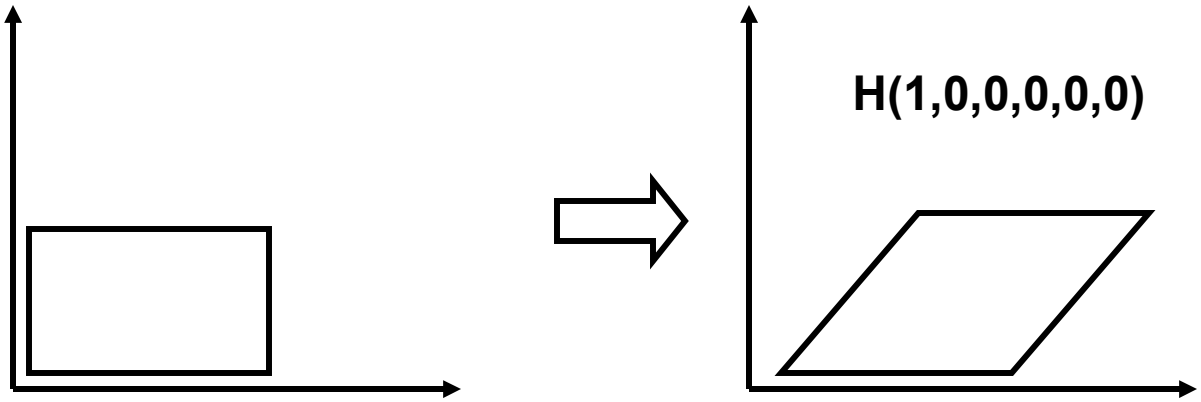
  - Warning: Change of orientation !



Basic Transformations

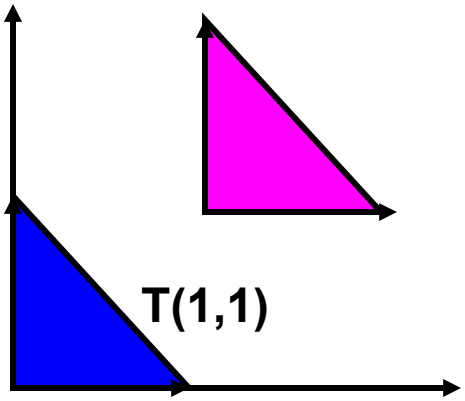
- **Shear (deutsch: Scherung)**

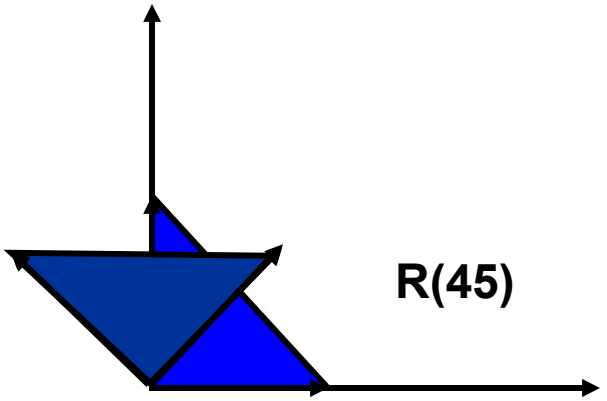
$$H(h_{xy}, h_{xz}, h_{yz}, h_{yx}, h_{zx}, h_{zy}) = \begin{pmatrix} 1 & h_{xy} & h_{xz} & 0 \\ h_{yx} & 1 & h_{yz} & 0 \\ h_{zx} & h_{zy} & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

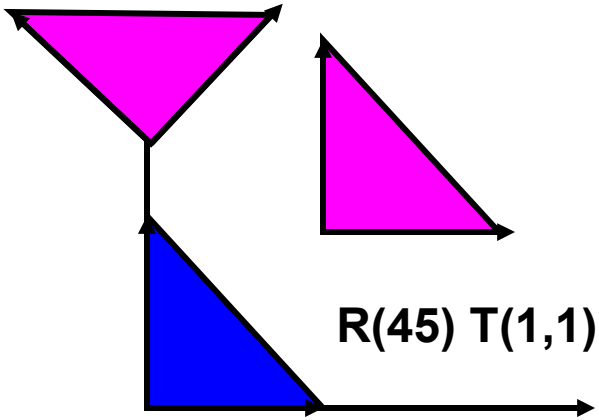


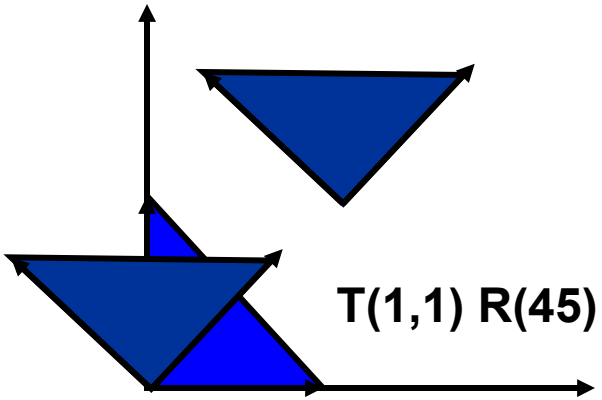
Concatenation of Transformations

- **In general, transformations do not commute**









Rotation About Arbitrary Points

- **Example: Rotate object about one vertex p**
  - Translate object from point to origin
  - Apply desired rotation
  - Translate object back to original position

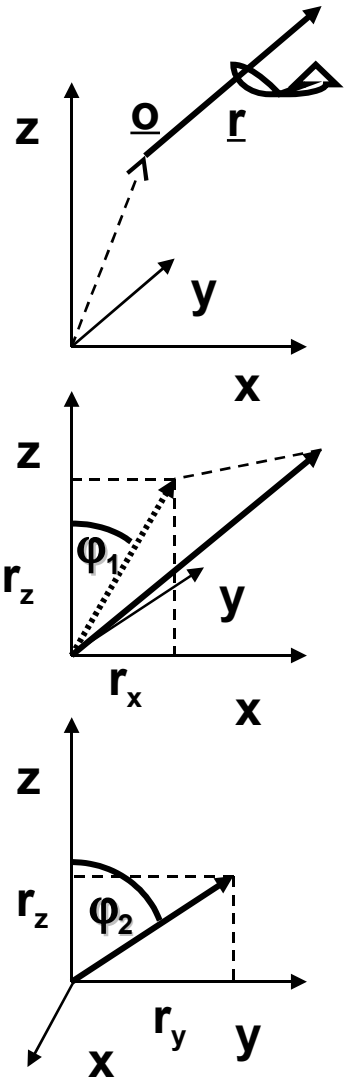
$$\boldsymbol{R}_a^p(\theta) = \boldsymbol{T}_{-p}\,\boldsymbol{R}_a(\theta)\,\boldsymbol{T}_p = \begin{bmatrix} 1 & 0 & 0 & p_x \\ 0 & 1 & 0 & p_y \\ 0 & 0 & 1 & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -p_x \\ 0 & 1 & 0 & -p_y \\ 0 & 0 & 1 & -p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

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

Rotation about arbitrary Axis

- Move base point to origin
  - $T(-\underline{o})$
- Rotation around Y-axis, so that $\underline{r}$ is in YZ-plane
  - Use projection into XZ-plane
  - $R_y(-\varphi_1)$ $\tan(\varphi_1)= r_x/r_z$ $r'= R_y(-\varphi_1)\, r$
- Rotation around X-axis, so that $\underline{r}'$ is along Z-axis
  - $R_x(\varphi_2)$ $\tan(\varphi_2)= r'_y/ r_z'$
- Rotation around Z-axis with angle φ
- Rotate back around X-axis
- Rotate back around Y-axis
- Translate back

- **Together**
  - $R(\varphi, \underline{o}, \underline{r}) = T(\underline{o})R_y(\varphi_1)R_x(-\varphi_2)R_z(\varphi)R_x(\varphi_2)R_y(-\varphi_1)T(-\underline{o})$



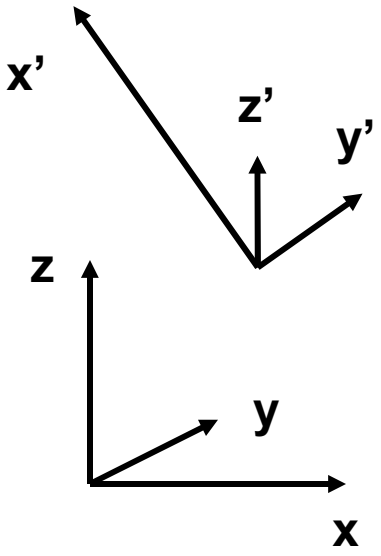
Interpretation of Matrices

- **Columns are transformed basis**

$$p' = \begin{pmatrix} t_{xx} & t_{xy} & t_{xz} & t_{xw} \\ t_{yx} & t_{yy} & t_{yz} & t_{yw} \\ t_{zx} & t_{zy} & t_{zz} & t_{zw} \\ t_{wx} & t_{wy} & t_{wz} & t_{ww} \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix} \text{oder,} \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} \underline{e}_x' & \underline{e}_y' & \underline{e}_z' & \underline{o}' \end{pmatrix} = M$$

- **Transformation into new basis**
  - Simple: Write new basis vectors into columns of matrix



Orthonormal Matrices

- **Orthonormal transformations**
  - Images of basis vectors are again orthonormal
    - $e_i' \cdot e_j' = \delta_{ij}$

$$\mathbf{M}^T\mathbf{M} = \begin{pmatrix} \underline{e}_x' & \underline{e}_y' & \underline{e}_z' \end{pmatrix}^T \begin{pmatrix} \underline{e}_x' & \underline{e}_y' & \underline{e}_z' \end{pmatrix} =$$

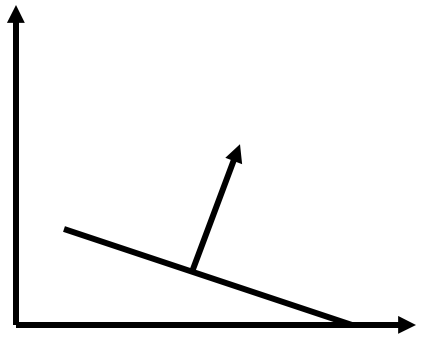
$$= \begin{pmatrix} \underline{e}_x'\underline{e}_x' & \underline{e}_x'\underline{e}_y' & \underline{e}_x'\underline{e}_z' \\ \underline{e}_y'\underline{e}_x' & \underline{e}_y'\underline{e}_y' & \underline{e}_y'\underline{e}_z' \\ \underline{e}_z'\underline{e}_x' & \underline{e}_z'\underline{e}_y' & \underline{e}_z'\underline{e}_z' \end{pmatrix} = \mathbf{1}$$

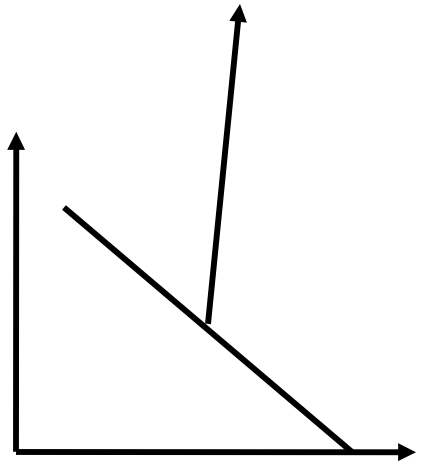
Which means that

$$\mathbf{M}^T = \mathbf{M}^{-1}$$

Transformations

- **Line**
  - Transform end points
- **Plane**
  - Transform three points
- **Vector**
  - $\underline{v} = \underline{p} - \underline{q} = (x, z, y, 0)^T$
  - Translations to not act on vectors
- **Normal vectors**
  - Problem: e.g. with non-uniform scaling

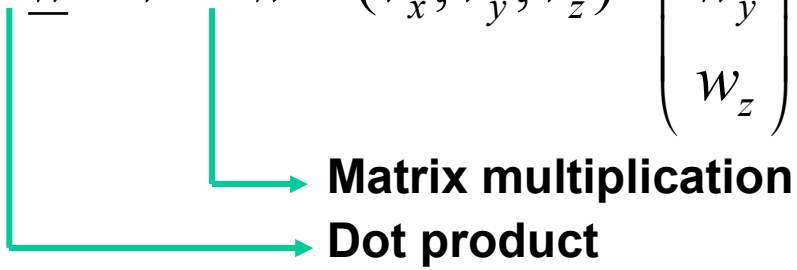




Transforming Normals

- **Dot product as matrix multiplication**

$$\underline{v} \cdot \underline{w} = v^T * w = (v_x, v_y, v_z) * \begin{pmatrix} w_x \\ w_y \\ w_z \end{pmatrix}$$



- **Normal N on a plane**
  - For any vector T in the plane: $\underline{N}^T * \underline{T} = 0$
  - Find transformation M' for normal vector, such that

$$(\mathbf{M}' * \underline{N})^T * (\mathbf{M} * \underline{T}) = 0 = \underline{N}^T * (\mathbf{M}'^T * \mathbf{M}) * \underline{T}$$

$$\mathbf{M}'^T * \mathbf{M} = 1$$

$$\mathbf{M}' = (\mathbf{M}^{-1})^T$$

Transforming Normals

- **Remember:**

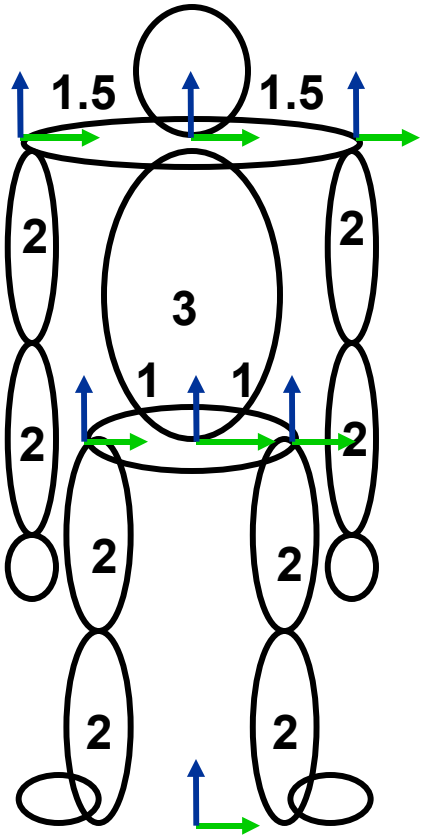
  Normals are transformed by the transpose of the inverse of the 4x4 transformation matrix of points and vectors

- **No problem with orthogonal transformations**
  - E.g. rotation, uniform scaling
    - $M^{-1} = M^{T}$
    - $M^{-1T} = M^{TT} = M$

Coordinate Systems

- **Object Coordinates**
  - Intrinsic coordinate system of an object
  - Hierarchical modeling
  - Modeling Transformations to world coordinates
- **World Coordinates**
  - Root for hierarchical modeling
  - Reference system for the camera
  - Viewing Transformation to camera coordinates
- **Camera coordinates (Viewing Coordinates)**
  - Reference system for lighting computations
  - Perspective Transformation to normalized (projection) coordinates

Hierarchical Coordinate Systems




```
body
   torso
      head
      shoulder
         larm
            upperarm
            lowerarm
            hand
         rarm
            upperarm
            lowerarm
            hand
      hips
         lleg
            upperleg
            lowerleg
            foot
         rleg
            upperleg
            lowerleg
            foot
```

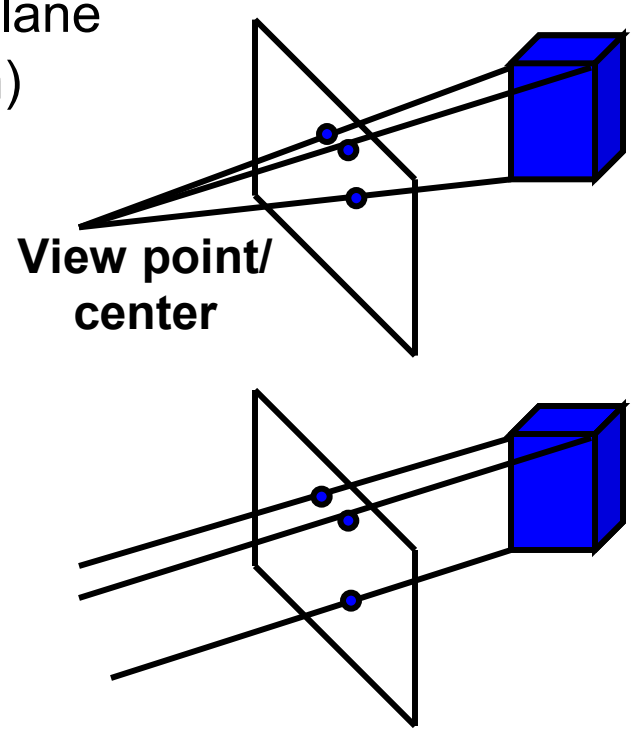
RenderMan Syntax

```
Translate 0 4 0
TransformBegin
  # Draw Torso
  Translate 0 3 0
  # Draw Shoulders
  TransformBegin
    Rotate a 0 0 1
    # Draw head
  TransformEnd
  TransformBegin
    Translate 1.5 0 0
    DRAW_ARM(a,b,c)
TransformEnd
TransformBegin
    Translate -1.5 0 0
    DRAW_ARM(d,e,f)
  TransformEnd
TransformEnd
# Draw hips
TransformBegin
  TransformBegin
    Translate 1 0 0
    DRAW_LEG(g,h)
TransformEnd
  TransformBegin
    Translate -1 0 0
    DRAW_LEG(i,j)
  TransformEnd
TransformEnd
```

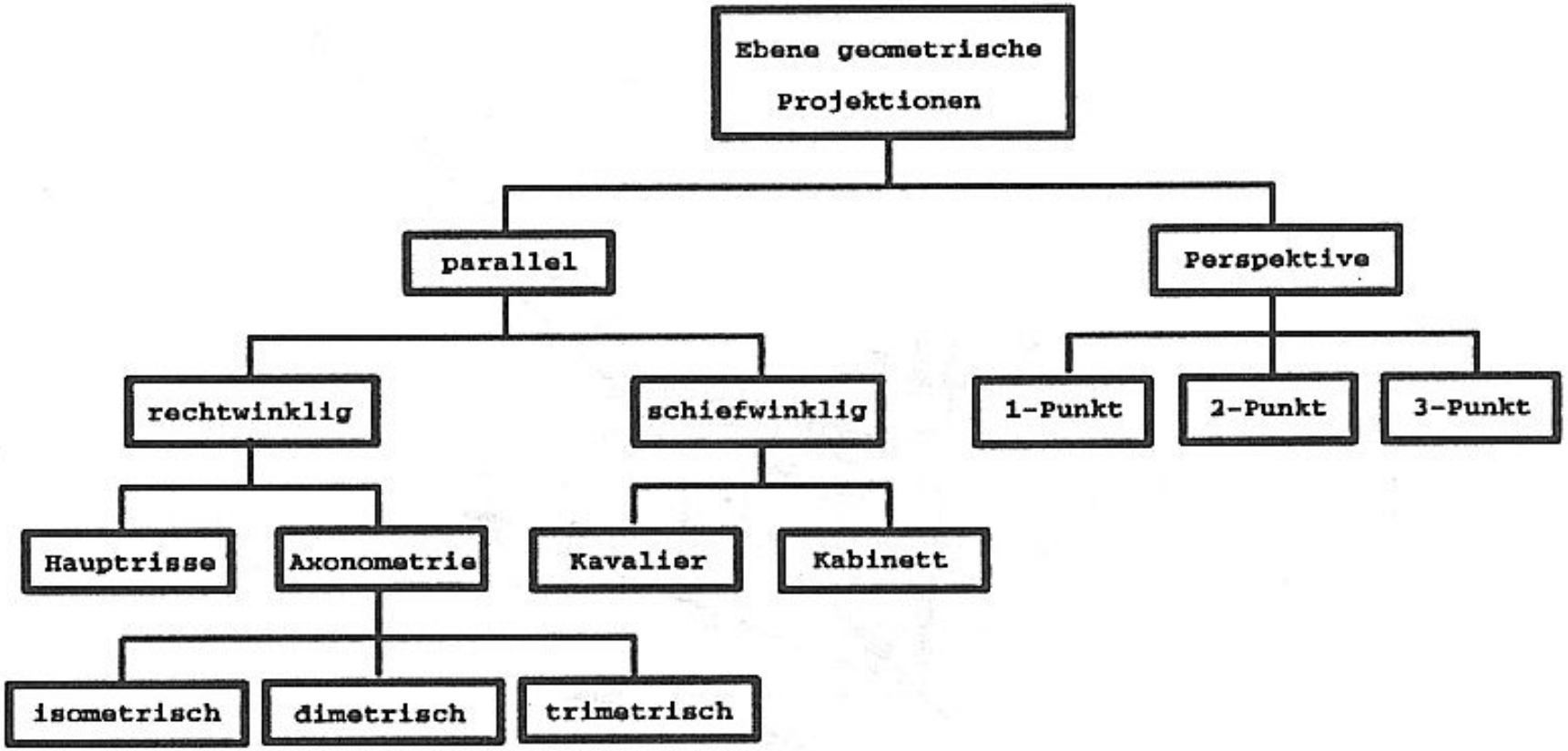
```
DRAW_ARM(a,b,c) {
      Rotate b 0 0 1
      # Draw upperarm
      Translate 0 -2 0
      Rotate c 1 0 0
      # Draw lowerarm
      Translate 0 -2 0
      # Draw hand
     }

DRAW_LEG(g,h) {
    Rotate g 1 0 0
    # Draw upperleg
    Translate 0 -2 0
    Rotate h 1 0 0
    # Draw lowerleg
    Translate 0 -2 0
    # Draw foot
    }
```

Projections

- **Definition: Projection**
  - Mapping from 3D to 2D
  - Results in loss of information
  - Non invertible
- **Planar perspective projections**
  - Projection along lines onto a projection plane
  - Perspective projection (central projection)
    - Lines intersect in a single point
  - Special case: Orthographic projection
    - Parallel lines
    - View point at infinity

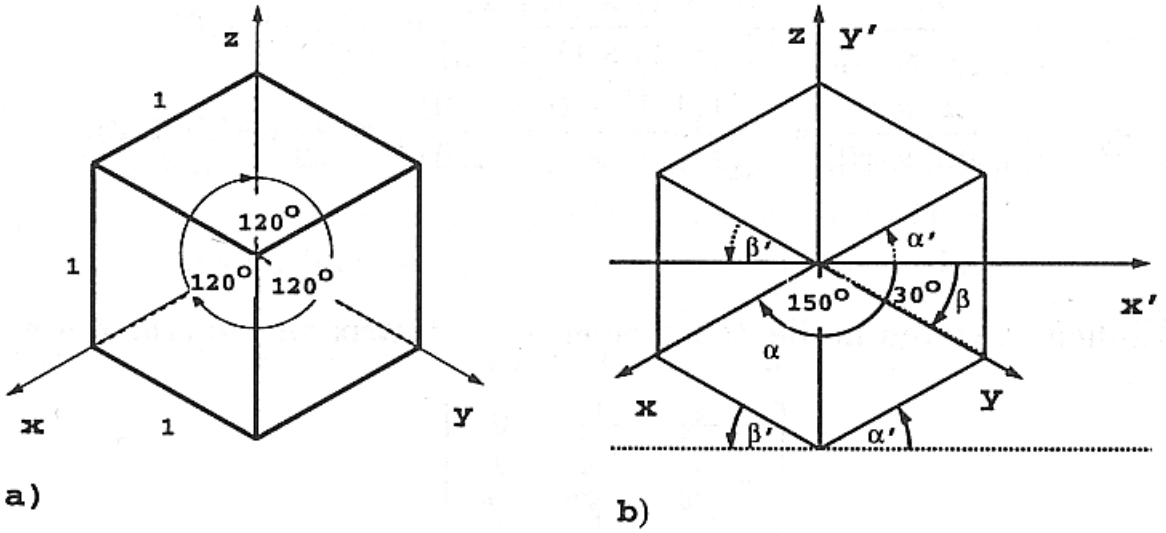


Classification of Projections



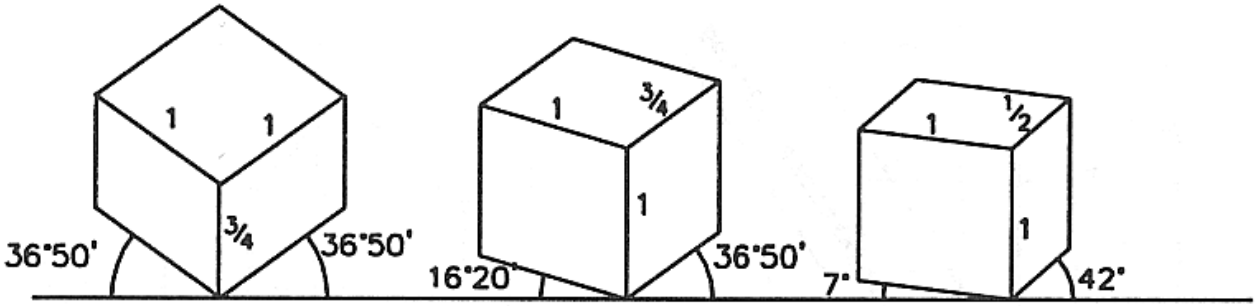
Axonometric Projection

- **Properties**
  - Parallel/orthographic projection
  - Projection plane orthogonal to projection direction
- **Isometric Projection**
  - Projektion direction has same angle with every coordinate axis
    - Lengths are maintained

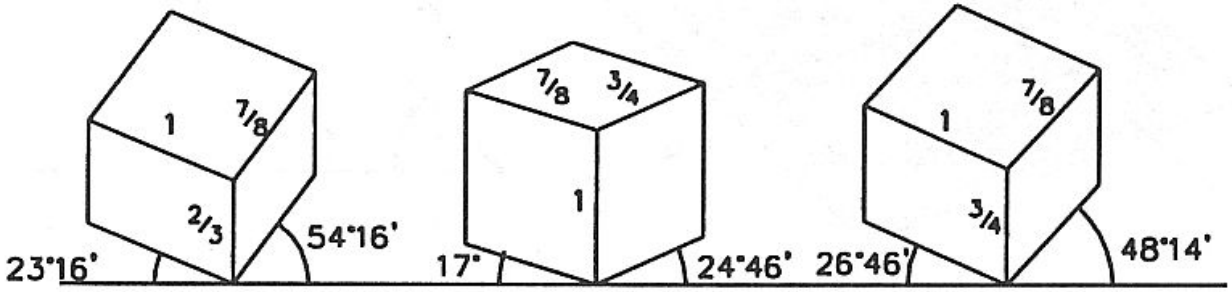


Axonometric Projection

- **Dimetric and Trimetric Projection**
  - Same angle with 2 axes
    - Two lengths are maintained, one is scaled

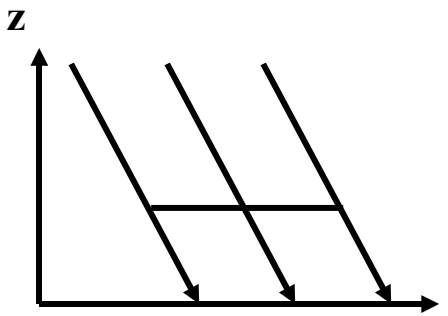


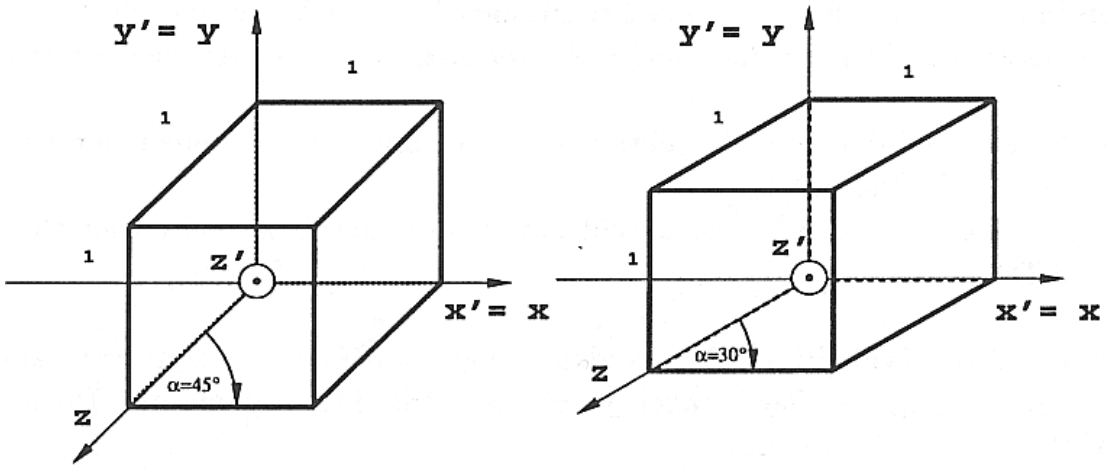
  - Same angle with one axis
    - One length is maintained, two are scaled



Oblique Projection

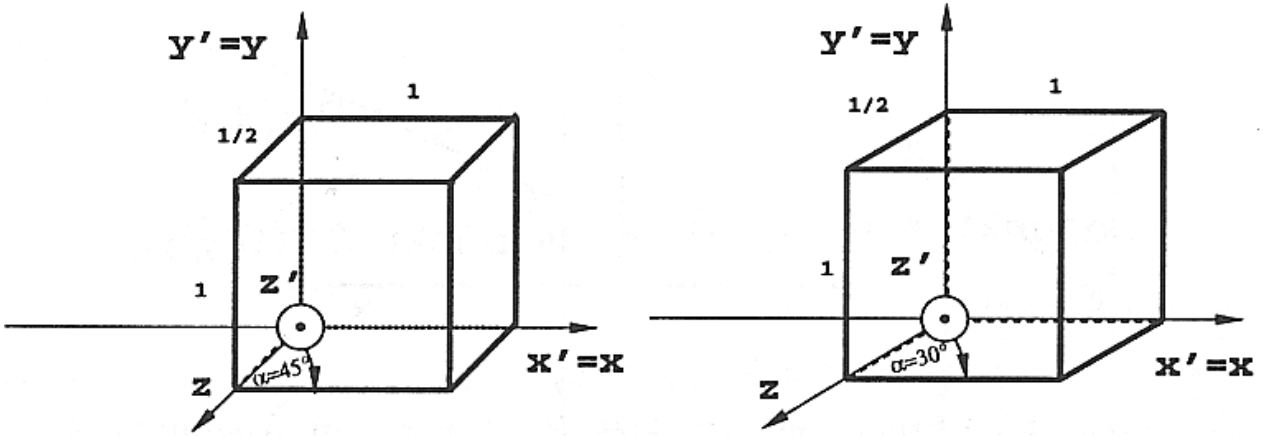
- **Properties**
  - Parallel projection
  - Projektion plane parallel to two coordinate axes (e.g. x, y)
  - Projection direction ***not*** orthogonal to plane
- **Cavalier Projection**
  - Same length on all axes



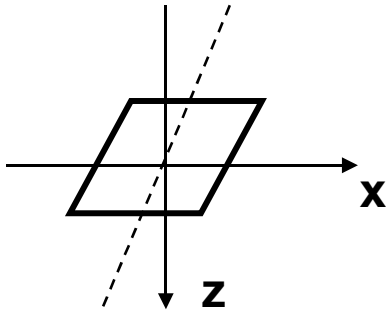


Oblique Projection

- **Cabinet Projection**
  - Foreshortening of ½ orthogonal to projection plane

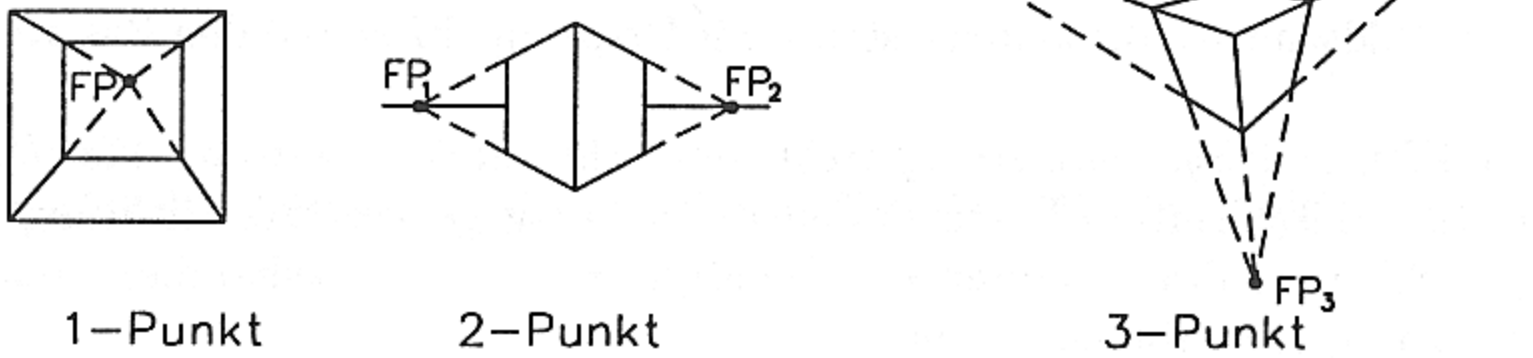


- **Implementation of Oblique Projections**
  - Shearing plus parallel projection

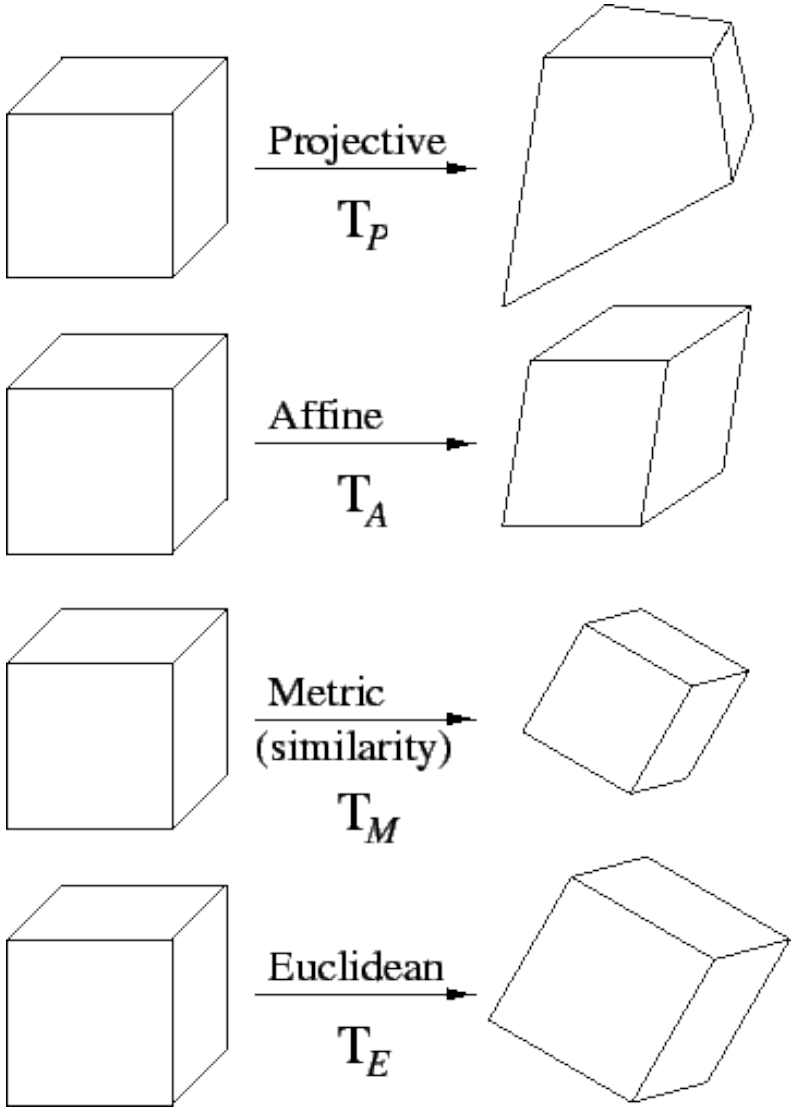


Planar Perspective Projection

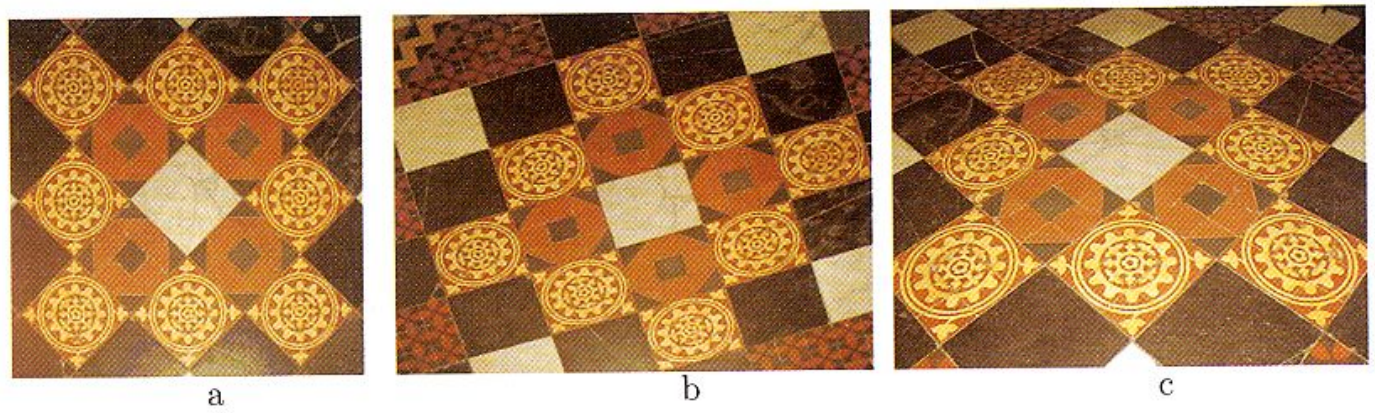
- **Properties**
  - Projection onto plane along lines through a projection point
  - Parallel lines do ***NOT*** stay parallel
    - Not an affine transformation
- **Vanishing point**
  - Projections of intersection points axis-parallel lines at infinity
  - N-point perspective
- **Details later**



Projective Transformations in 3D



Distortions under Central Projection



- **Similarity**: circle remains circle, square remains square

⇒ line orientation is preserved

- **Affine**: circle becomes ellipse, square becomes rhombus

⇒ parallel lines remain parallel

- **Projective**: imaged object size depends on distance from camera

⇒ parallel lines converge