# Computer Graphics 1

## Ludwig-Maximilians-Universität München

Summer semester 2020


## Prof. Dr.-Ing. Andreas Butz

lecture additions by Dr. Michael Krone, Univ. Stuttgart


http://www.wikiwand.com/

# Chapter 2 – Transformations & Scene Graphs

- Three-Dimensional Geometric Transformations
- Affine Transformations and Homogeneous Coordinates
- Combining Transformations

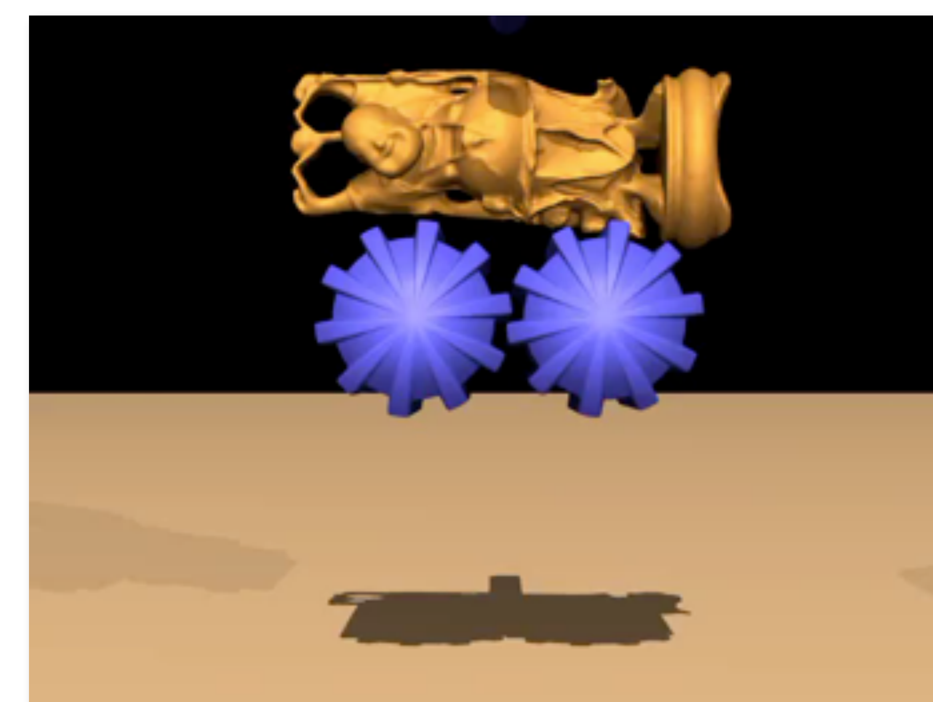
- Why a scene graph?
- What is stored in the scene graph?
  - Objects
  - Appearance
  - Camera
  - Lights
- Rendering with a scene graph
- Practical example

# What is a Transformation?

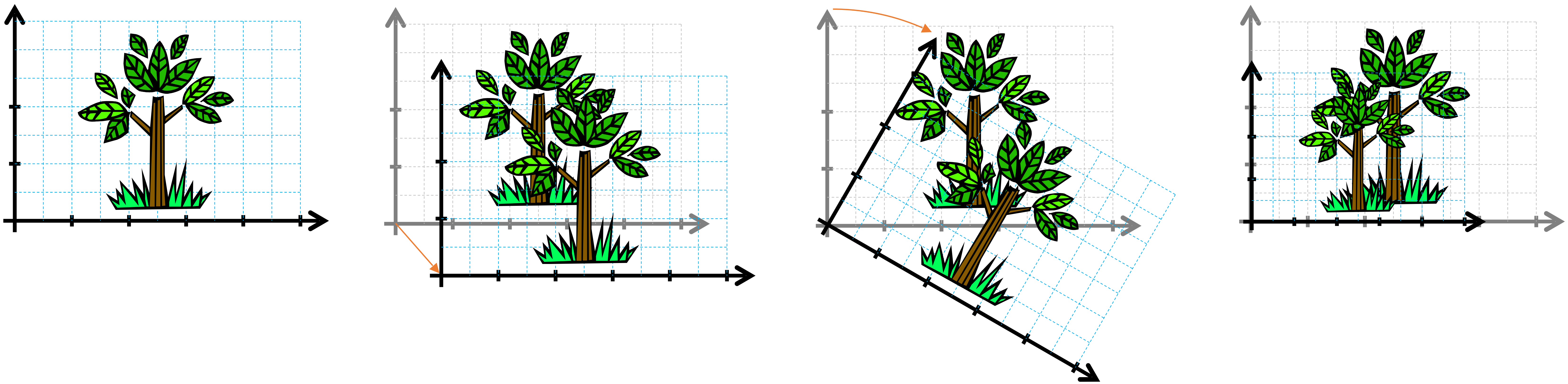
- A transformations maps a point $x$ to a point $x'$

- A linear transformation $T$ is a mapping that maps a point $x \in \mathbb{R}^n$ to a point $x' \in \mathbb{R}^m$: $T(x) = Ax$
  - $A \in \mathbb{R}^{m \times n}$ is the transformation matrix of $T$
  - $A$ has $m$ rows and $n$ columns

- Examples
  - Positioning of an object in the scene
  - Animation
  - Deformation
  - Camera transformations and projection (← later!)
  - But also real-time shadows, mirroring etc.



# Basic Transformations



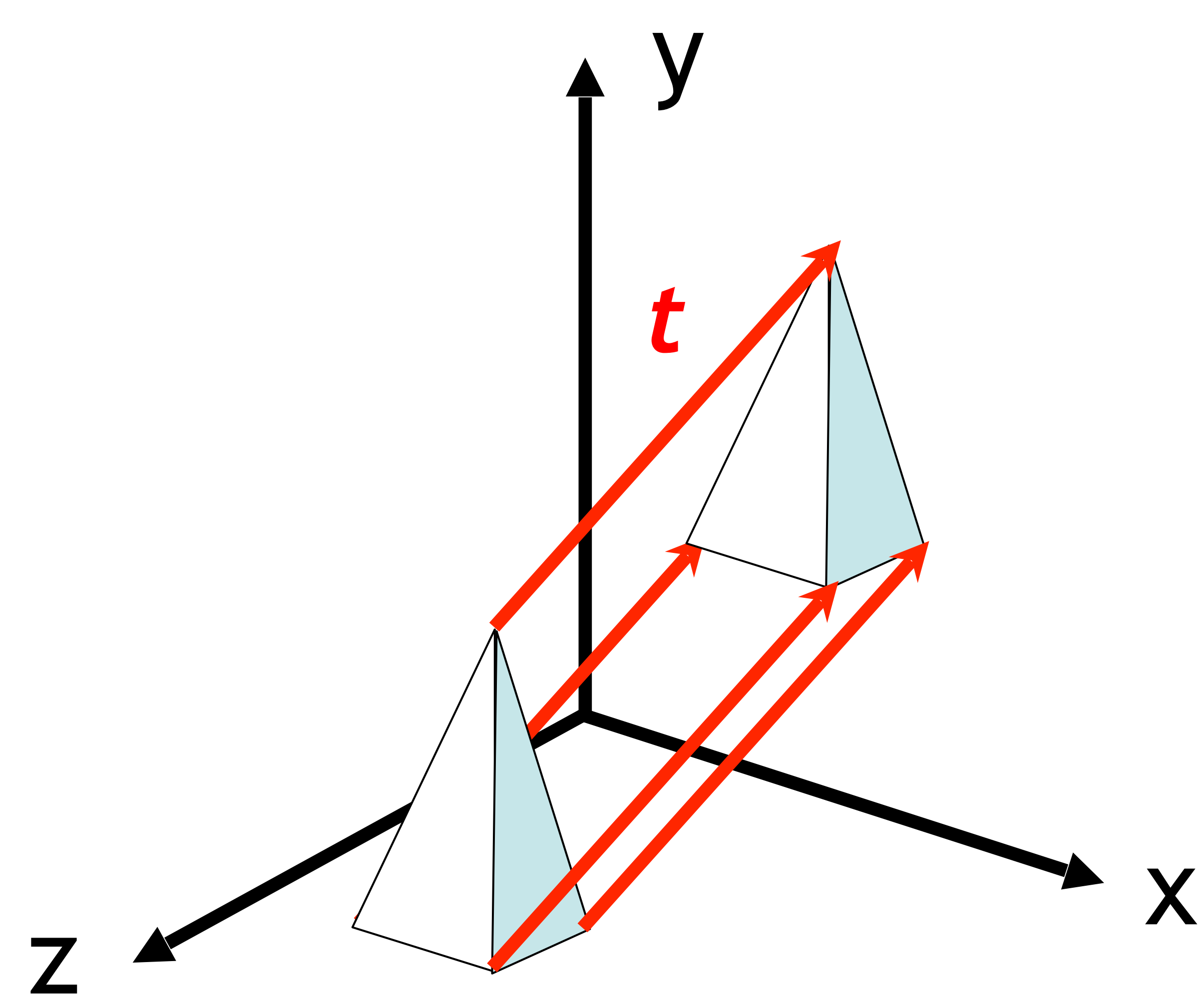
Identity | Translation | Rotation | isotrope (uniform) Scaling

- Transformations in CG normally…
  - can be combined and…
  - are reversible/invertible
    - Exception: Scaling by a factor of zero!

# Translation

- Add a vector $t$
- Geometrical meaning: Shifting
- Inverse operation?
- Neutral operation?

$$\begin{pmatrix} p_1 \\ p_2 \\ p_3 \end{pmatrix} + \begin{pmatrix} t_1 \\ t_2 \\ t_3 \end{pmatrix} = \begin{pmatrix} p_1 + t_1 \\ p_2 + t_2 \\ p_3 + t_3 \end{pmatrix}$$

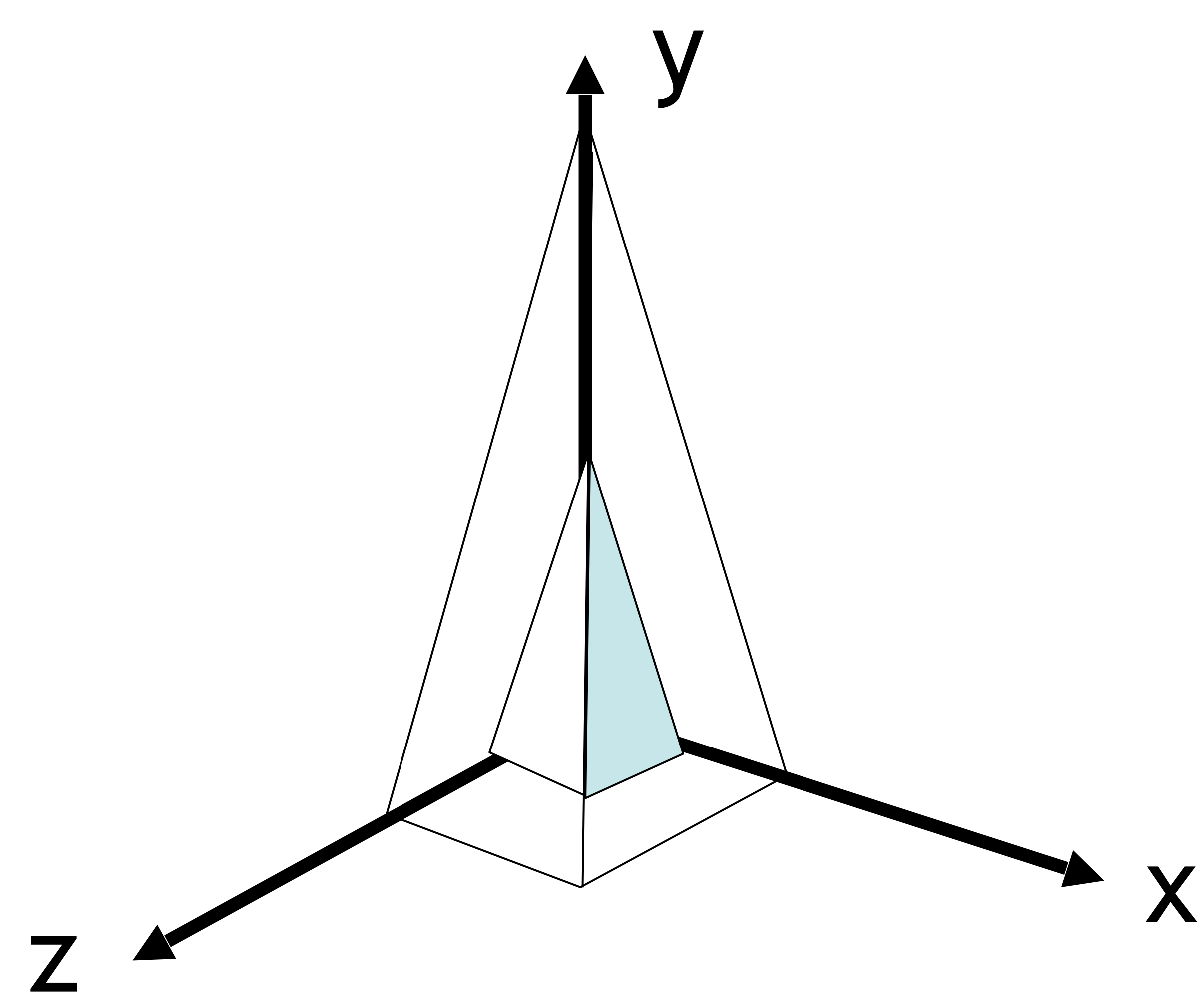


# Uniform Scaling

- Multiply with a scalar *s*
- Geometrical meaning: Changing the size of an object

$$\begin{pmatrix} p_1 \\ p_2 \\ p_3 \end{pmatrix} \cdot s = \begin{pmatrix} p_1 \cdot s \\ p_2 \cdot s \\ p_3 \cdot s \end{pmatrix}$$

- What happens when we scale objects which are not at the origin?
- How can we fix that?

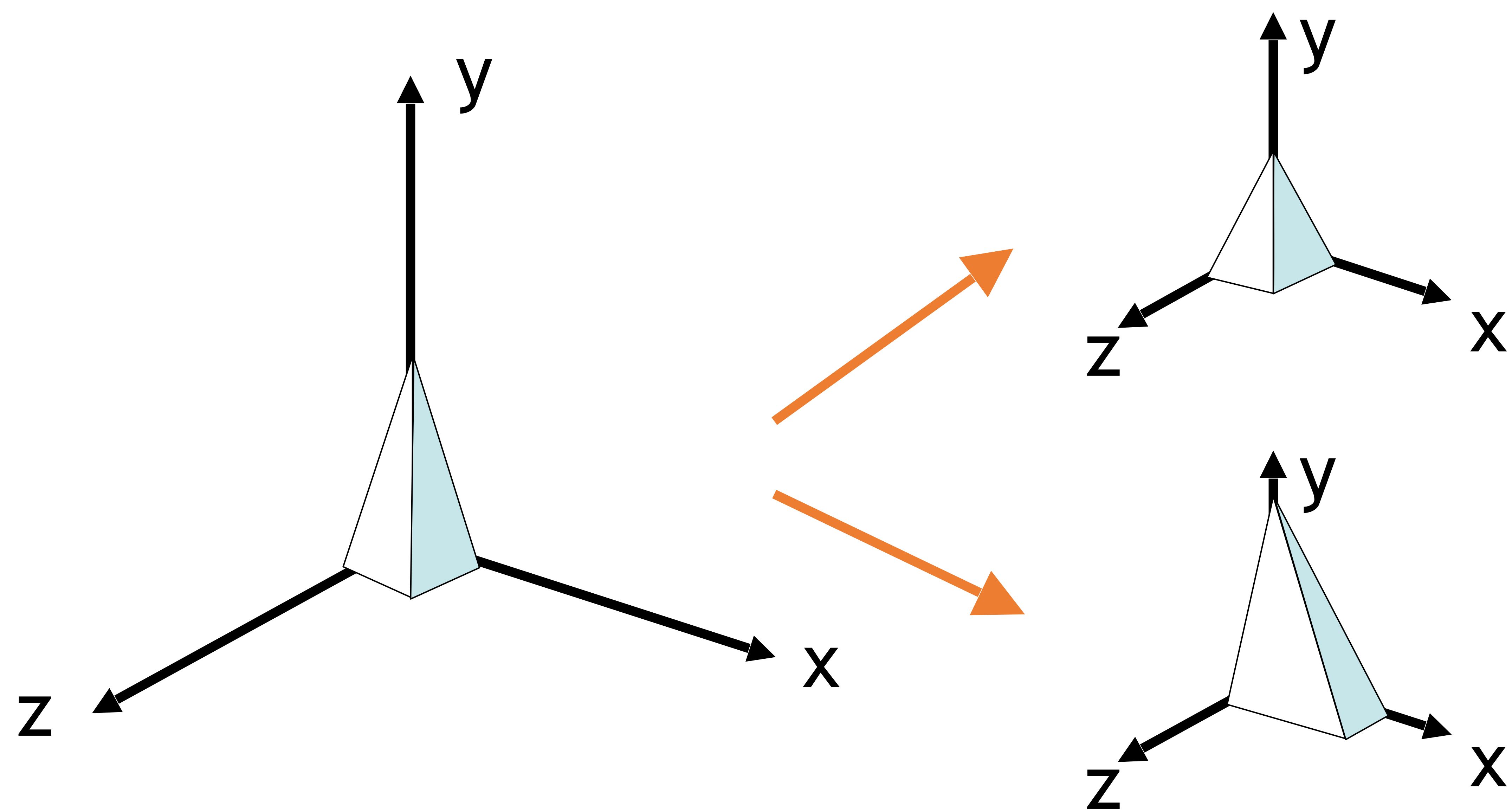


# Non-Uniform Scaling

- Multiply with three scalars
- One for each dimension
- Geometrical meaning?

$$\begin{pmatrix} s_1 & 0 & 0 \\ 0 & s_2 & 0 \\ 0 & 0 & S_3 \end{pmatrix} \cdot \begin{pmatrix} p_1 \\ p_2 \\ p_3 \end{pmatrix} = \begin{pmatrix} p_1 \cdot s_1 \\ p_2 \cdot s_2 \\ p_3 \cdot s_3 \end{pmatrix}$$

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



# Reflection (Mirroring)

- Special case of scaling

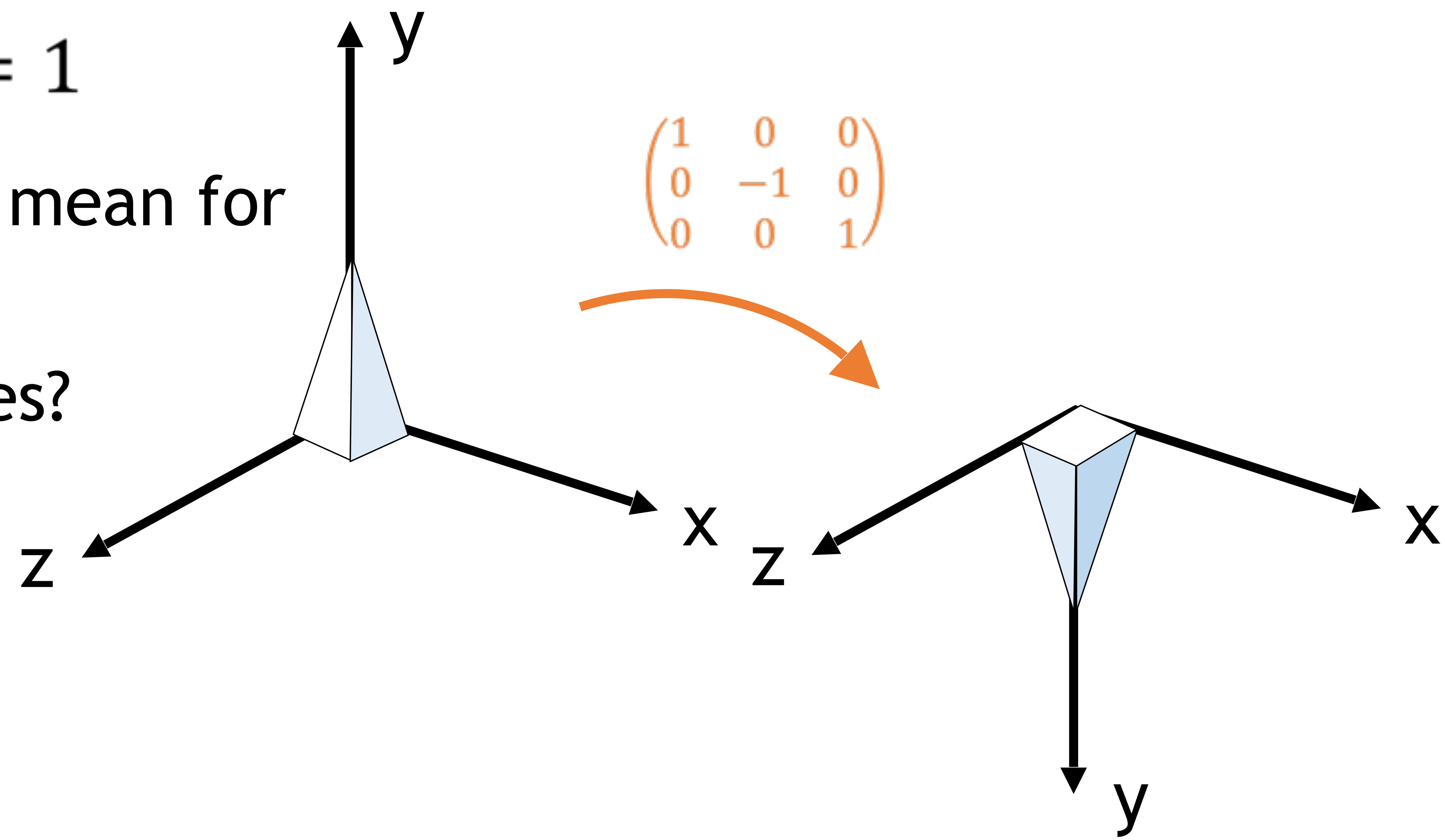
  $$s_1 \cdot s_2 \cdot s_3 < 0$$

$$\begin{pmatrix} s_1 & 0 & 0 \\ 0 & s_2 & 0 \\ 0 & 0 & s_3 \end{pmatrix} \cdot \begin{pmatrix} p_1 \\ p_2 \\ p_3 \end{pmatrix} = \begin{pmatrix} p_1 \cdot s_1 \\ p_2 \cdot s_2 \\ p_3 \cdot s_3 \end{pmatrix}$$

- Example:

  $$s_1 = 1, s_2 = -1, s_3 = 1$$

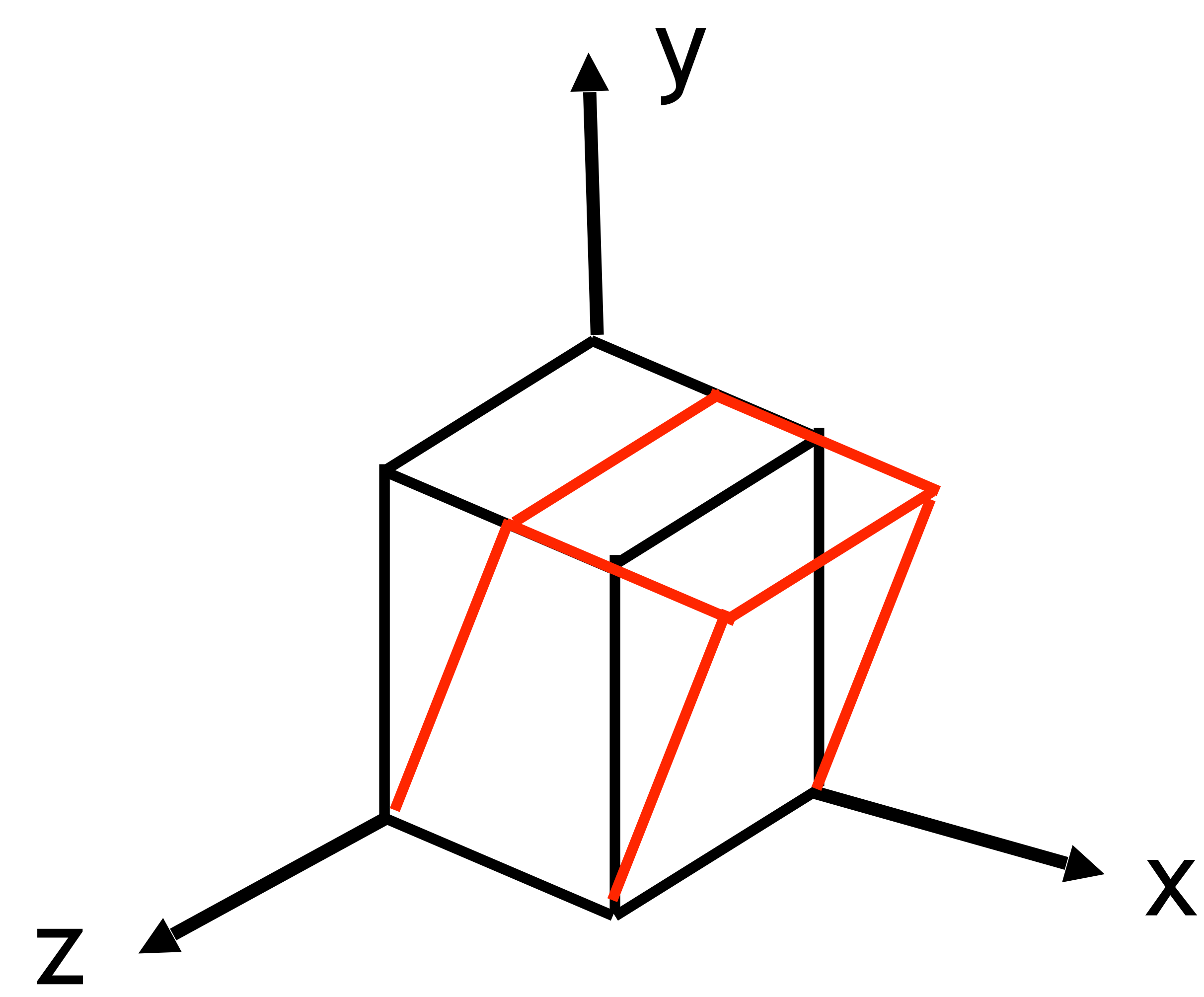
- Discuss: What does this mean for
  - surface normals?
  - order of polygon edges?
  - handedness?



# Shearing along X Axis

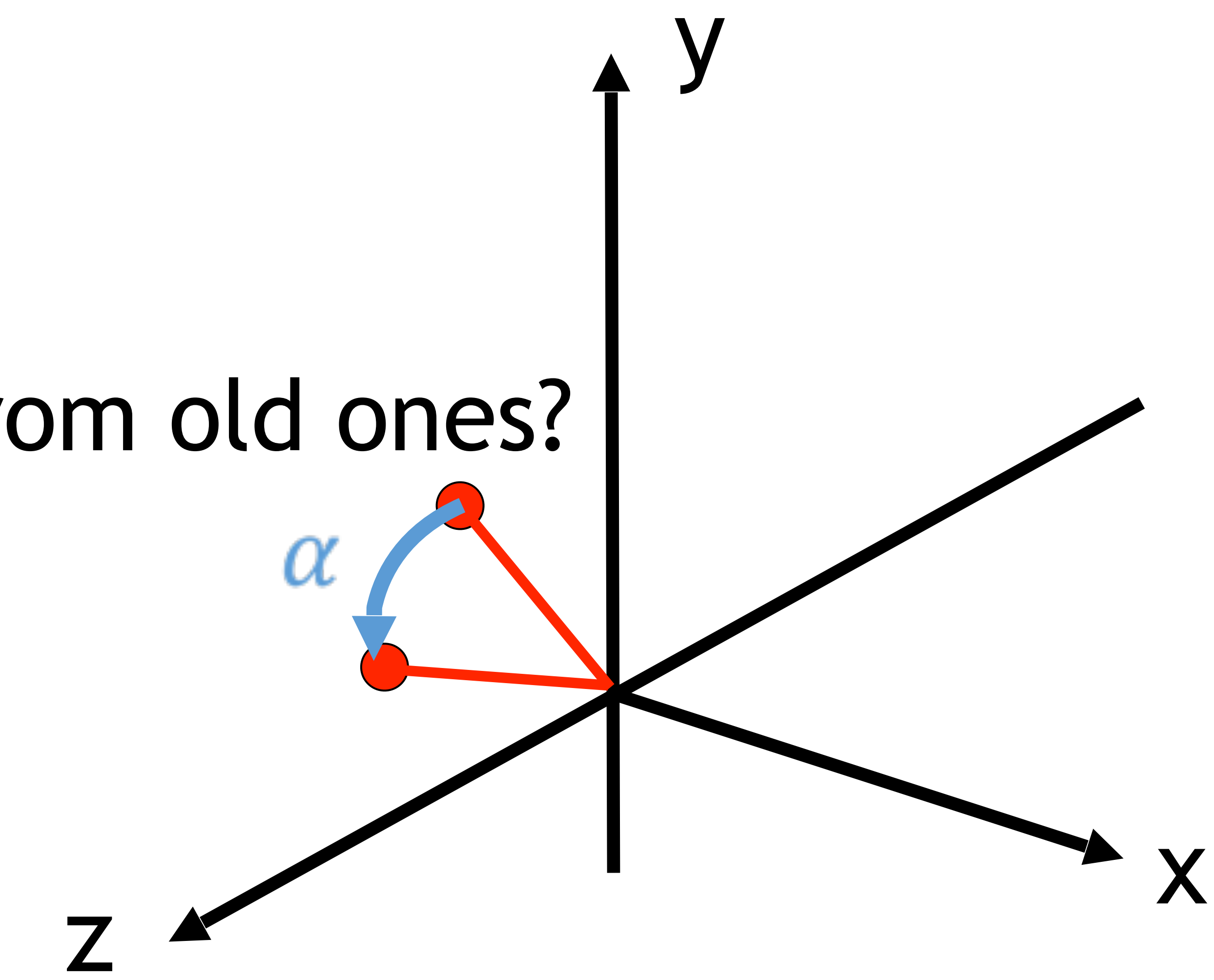
- Example:
  - Only x coordinate values are modified
  - Modification depends linearly on y coordinate value
  - Areas in x/y and x/z plane, as well as volume remain the same
- Generalization to other axes and arbitrary axis: see later...

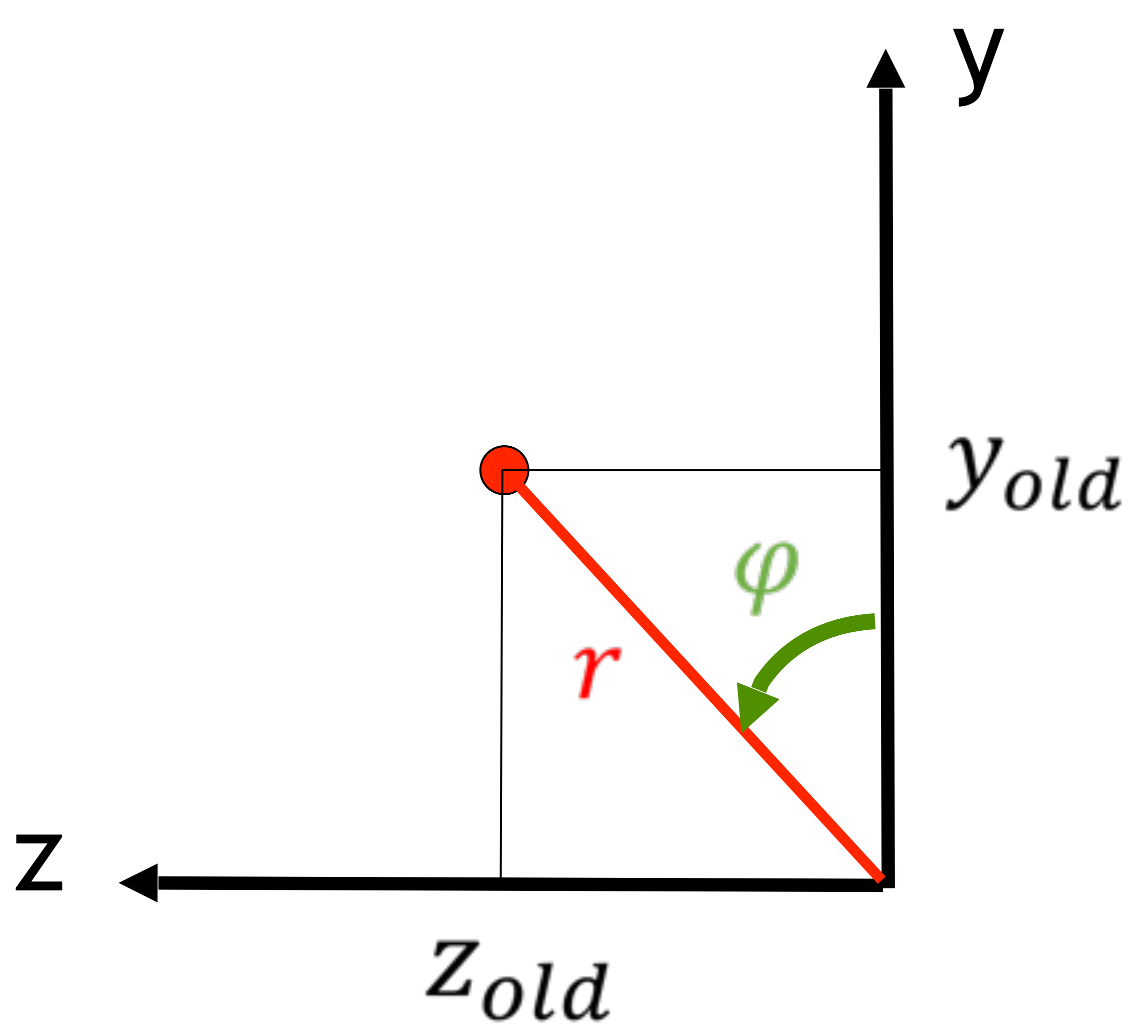
$$\begin{pmatrix} p_1 + m \cdot p_2 \\ p_2 \\ p_3 \end{pmatrix} = \begin{pmatrix} 1 & m & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} p_1 \\ p_2 \\ p_3 \end{pmatrix}$$



# Rotation about X Axis (1/3)

- x coordinate value remains constant
- Rotation takes place in y/z-plane (2D)
- How to compute new y and z coordinates from old ones?

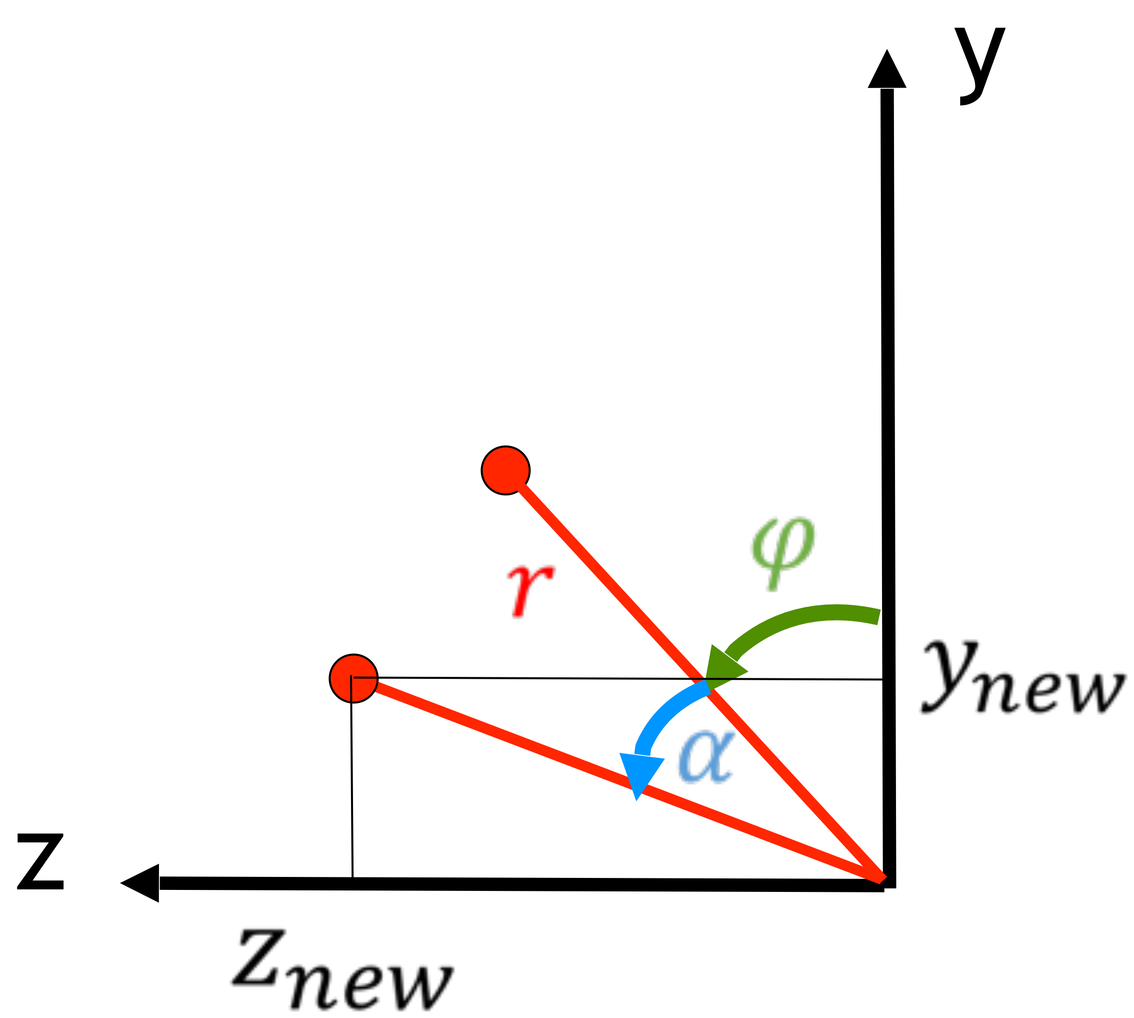



$$\cos\varphi = \frac{y_{old}}{r} \qquad y_{old} = r \cdot \cos\varphi$$

$$\sin\varphi = \frac{z_{old}}{r} \qquad z_{old} = r \cdot \sin\varphi$$

# Rotation about X Axis (2/3)




$$y_{old} = r \cdot \cos\varphi$$

$$z_{old} = r \cdot \sin\varphi$$

$$\cos(\alpha + \varphi) = \frac{y_{new}}{r}$$

$$\begin{aligned} y_{new} &= r \cdot \cos(\alpha + \varphi) \\ &= r \cdot \cos\alpha \cdot \cos\varphi - r \cdot \sin\alpha \cdot \sin\varphi \\ &= \cos\alpha \cdot y_{old} - \sin\alpha \cdot z_{old} \end{aligned}$$

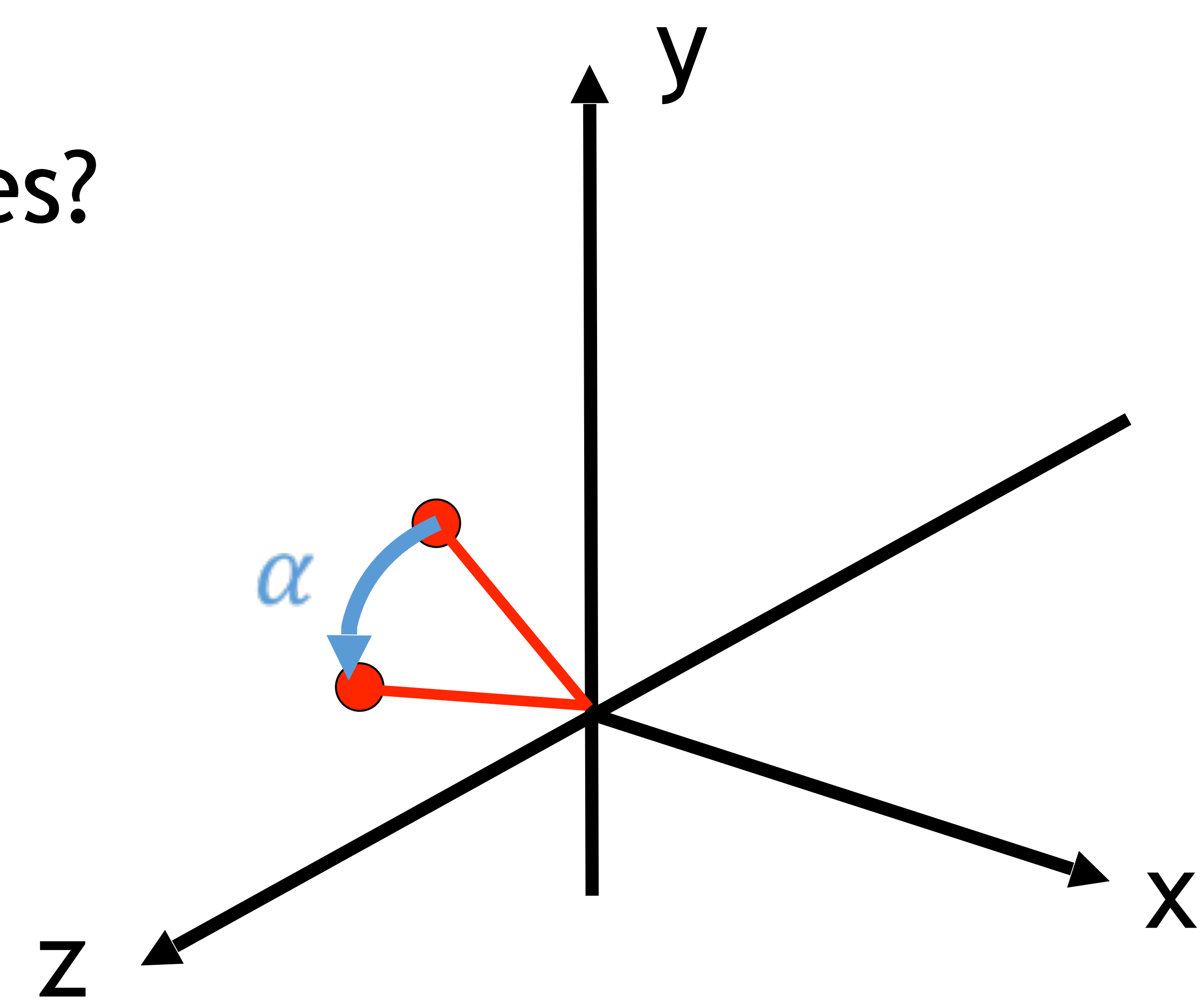
$$\sin(\alpha + \varphi) = \frac{z_{new}}{r}$$

$$\begin{aligned} z_{new} &= r \cdot \sin(\alpha + \varphi) \\ &= r \cdot \sin\alpha \cdot \cos\varphi + r \cdot \cos\alpha \cdot \sin\varphi \\ &= \sin\alpha \cdot y_{old} + \cos\alpha \cdot z_{old} \end{aligned}$$

# Rotation about X Axis (3/3)

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{pmatrix} \cdot \begin{pmatrix} p_1 \\ p_2 \\ p_3 \end{pmatrix} = \begin{pmatrix} p_1 \\ \cos\alpha \cdot p_2 - \sin\alpha \cdot p_3 \\ \sin\alpha \cdot p_2 + \cos\alpha \cdot p_3 \end{pmatrix}$$

- Special cases, e.g. 90 degrees, 180 degrees?
- How to rotate about other axes?



# Elementary rotations

- Combine to express arbitrary rotation
- This is not always intuitive
- Order matters (a lot!) → Likely source of mistakes/bugs!

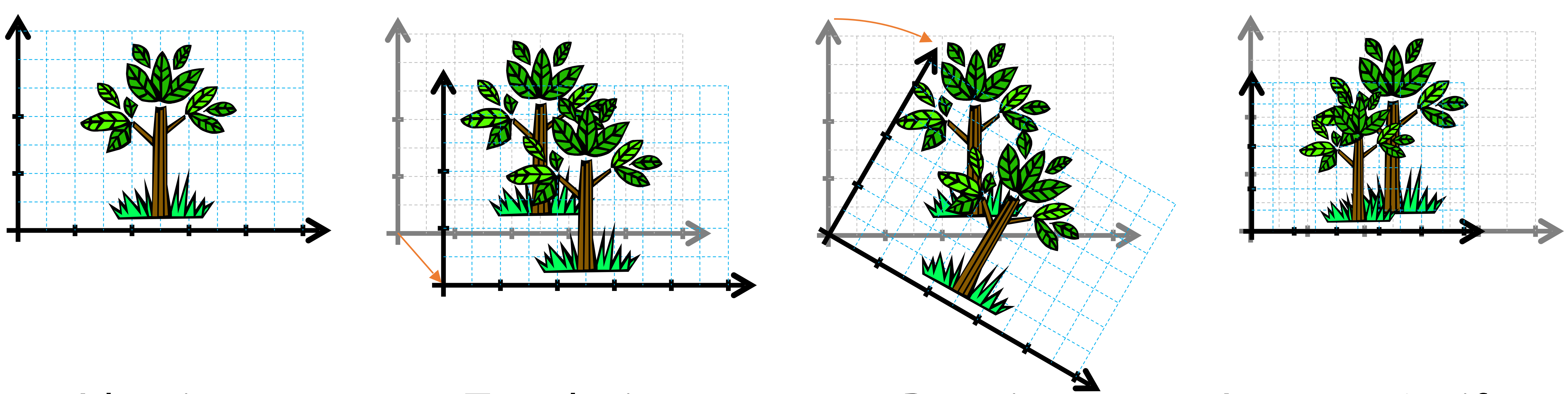
$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{pmatrix} \cdot \begin{pmatrix} p_1 \\ p_2 \\ p_3 \end{pmatrix} = \begin{pmatrix} p_1 \\ \cos\alpha \cdot p_2 - \sin\alpha \cdot p_3 \\ \sin\alpha \cdot p_2 + \cos\alpha \cdot p_3 \end{pmatrix}$$

$$\begin{pmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{pmatrix} \cdot \begin{pmatrix} p_1 \\ p_2 \\ p_3 \end{pmatrix} = \begin{pmatrix} \cos\beta \cdot p_1 + \sin\beta \cdot p_3 \\ p_2 \\ \cos\beta \cdot p_3 - \sin\beta \cdot p_1 \end{pmatrix}$$

$$\begin{pmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} p_1 \\ p_2 \\ p_3 \end{pmatrix} = \begin{pmatrix} \cos\gamma \cdot p_1 - \sin\gamma \cdot p_2 \\ \sin\gamma \cdot p_1 + \cos\gamma \cdot p_2 \\ p_3 \end{pmatrix}$$

# Transformation of Coordinate Systems

- Applying a geometric transformation...
  - ...to all points of a single object: Transforming the object within its own coordinate system.
  - ...to all points of all objects of the “world”: effectively transforming the reference coordinate system in the opposite direction!
- Geometric transformations can be used to...
  - …modify an object
  - ...place an object within a reference coordinate system
  - ...switch to different reference coordinates

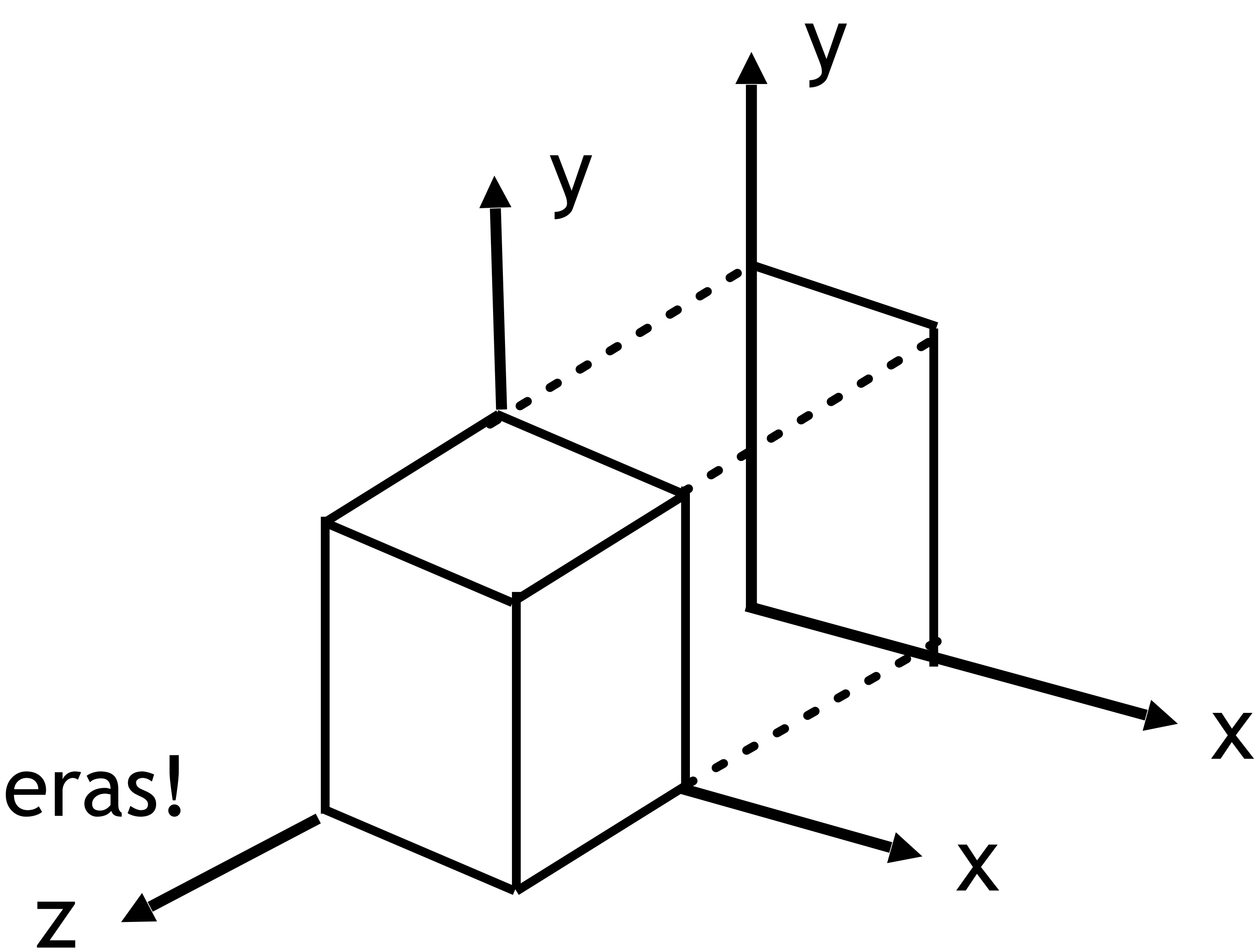


Identity — Translation — Rotation — Isotrope (uniform) scaling

# Transformation from 3D to 2D: Projection

- Many different projections exist (see later)
- Projection onto x/y plane:
  - “Forget” the z coordinate value

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} p_1 \\ p_2 \\ p_3 \end{pmatrix} = \begin{pmatrix} p_1 \\ p_2 \end{pmatrix}$$




- Other projections?
- Other viewpoints?

→ More detail in lecture about cameras!

# Chapter 2 – Transformations & Scene Graphs

- Three-Dimensional Geometric Transformations
- Affine Transformations and Homogeneous Coordinates
- Combining Transformations

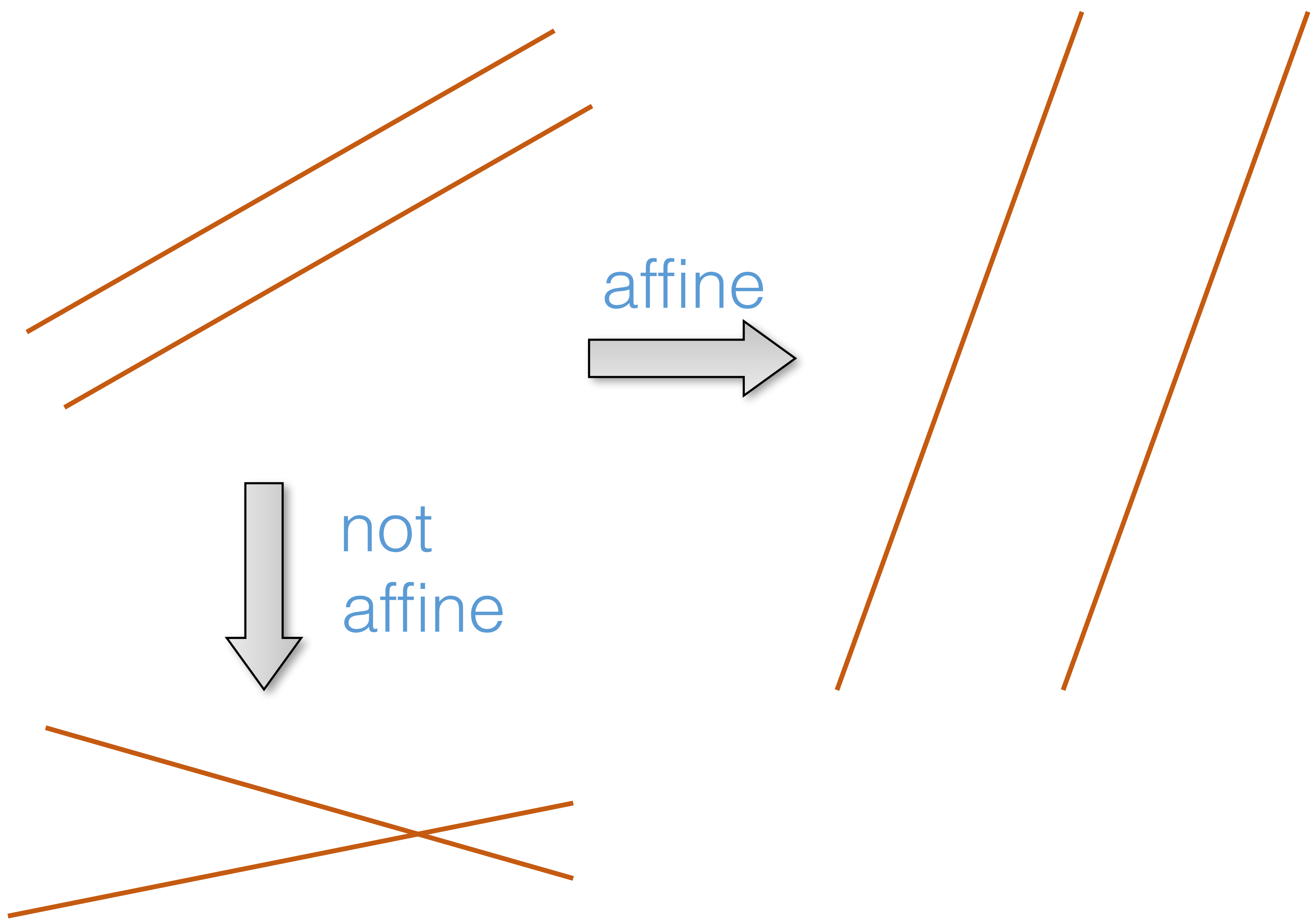
- Why a scene graph?
- What is stored in the scene graph?
  - Objects
  - Appearance
  - Camera
  - Lights
- Rendering with a scene graph
- Practical example

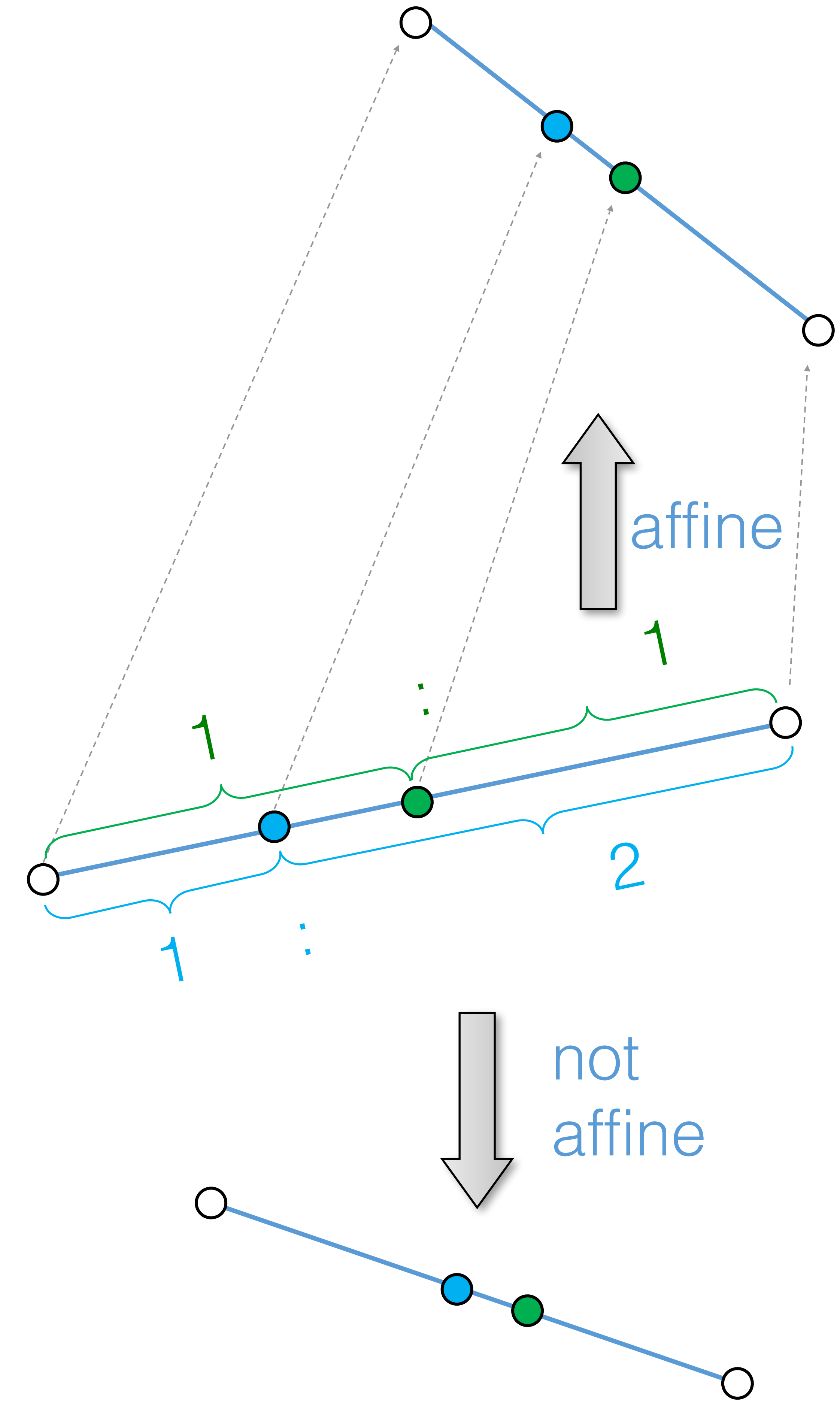
# Affine Transformations

- Mathematically: A transformation preserving collinearity
  - Points lying on a line before are on a line after transformation
  - Ratios of distances are preserved (e.g. midpoint of a line segment)
  - Parallel lines remain parallel
  - Angles, lengths and areas are *not* necessarily preserved!
- Basic transformations: translation, rotation, scaling and shearing
  - All combinations of these are affine transformations again
  - Combination is associative, but not commutative
- General form of computation:
  - New coordinate values are defined by linear function of the old values

$$\begin{pmatrix} p_1' \\ p_2' \\ p_3^1 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \cdot \begin{pmatrix} p_1 \\ p_2 \\ p_3 \end{pmatrix} + \begin{pmatrix} t_1 \\ t_2 \\ t_3 \end{pmatrix} = A \cdot p + t$$

# Affine Transformations



# Combining Multiple Transformations

- Rotation, scaling and shearing are expressed as matrices
  - Associative, hence can all be combined into one matrix
  - **Many** of these operations can also be combined into one matrix

- Translation is expressed by adding a vector
  - Adding vectors is also associative
  - Many translations can be combined into a single vector

- Combination of Translation with other operations?
  - Series of matrix multiplications and vector additions, difficult to combine
  - How about using a matrix multiplication to express translation?!?

# Homogeneous Coordinates

- Usage of a representation of coordinate-positions with an extra dimension
  - Extra value is a *scaling factor*
- 3D position ($x$, $y$, $z$) is represented by ($x_h$, $y_h$, $z_h$, $h$) such that

$$x = \frac{x_h}{h}, \qquad y = \frac{y_h}{h}, \qquad z = \frac{z_h}{h}$$

$$\begin{pmatrix} x_h \\ y_h \\ z_h \\ h \end{pmatrix}$$

- Simple choice for scaling factor $h$ is the value 1
  - In special cases other values can be used
- 3D position ($x$, $y$, $z$) is represented by ($x$, $y$, $z$, 1)
  - Position vector
- 3D direction ($x$, $y$, $z$) is represented by ($x$, $y$, $z$, 0)

# Translation Expressed in Homogeneous Coordinates

$$\begin{pmatrix} p_1' \\ p_2' \\ p_3' \end{pmatrix} = \begin{pmatrix} t_1 \\ t_2 \\ t_3 \end{pmatrix} + \begin{pmatrix} p_1 \\ p_2 \\ p_3 \end{pmatrix} = \begin{pmatrix} p_1 + t_1 \\ p_2 + t_2 \\ p_3 + t_3 \end{pmatrix}$$

$$\begin{pmatrix} p_1' \\ p_2' \\ p_3' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & t_1 \\ 0 & 1 & 0 & t_2 \\ 0 & 0 & 1 & t_3 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} p_1 \\ p_2 \\ p_3 \\ 1 \end{pmatrix} = \begin{pmatrix} p_1 + t_1 \\ p_2 + t_2 \\ p_3 + t_3 \\ 1 \end{pmatrix}$$

→ Translation has no effect on direction vector $(x, y, z, 0)$!

$$\begin{pmatrix} p_1' \\ p_2' \\ p_3' \\ 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & t_1 \\ 0 & 1 & 0 & t_2 \\ 0 & 0 & 1 & t_3 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} p_1 \\ p_2 \\ p_3 \\ 0 \end{pmatrix} = \begin{pmatrix} p_1 + 0 \cdot t_1 \\ p_2 + 0 \cdot t_2 \\ p_3 + 0 \cdot t_3 \\ 0 \cdot 1 \end{pmatrix}$$

# Scaling Expressed in Homogeneous Coordinates

$$\begin{pmatrix} p_1' \\ p_2' \\ p_3' \end{pmatrix} = \begin{pmatrix} s_1 & 0 & 0 \\ 0 & s_2 & 0 \\ 0 & 0 & s_3 \end{pmatrix} \cdot \begin{pmatrix} p_1 \\ p_2 \\ p_3 \end{pmatrix} = \begin{pmatrix} s_1 p_1 \\ s_2 p_2 \\ s_3 p_3 \end{pmatrix}$$

$$\begin{pmatrix} p_1' \\ p_2' \\ p_3' \\ 1 \end{pmatrix} = \begin{pmatrix} s_1 & 0 & 0 & 0 \\ 0 & s_2 & 0 & 0 \\ 0 & 0 & s_3 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} p_1 \\ p_2 \\ p_3 \\ 1 \end{pmatrix} = \begin{pmatrix} s_1 p_1 \\ s_2 p_2 \\ s_3 p_3 \\ 1 \end{pmatrix}$$

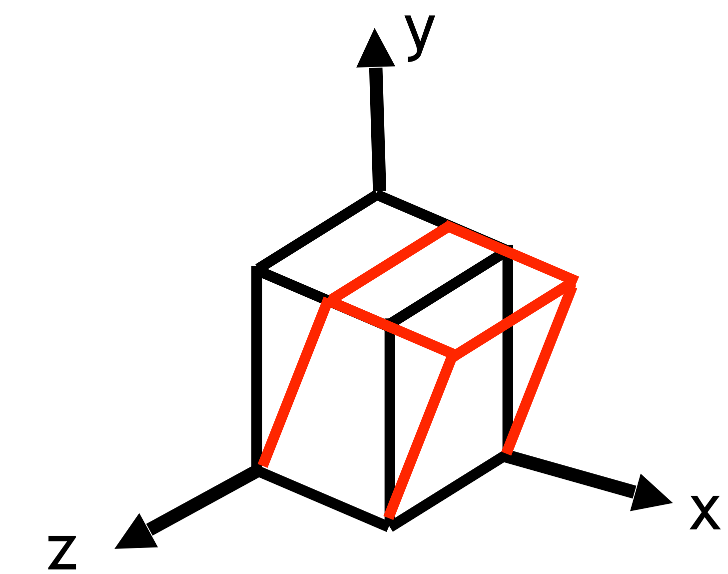
# Rotation Expressed in Homogeneous Coordinates

$$\begin{pmatrix} p_1' \\ p_2' \\ p_3' \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{pmatrix} \cdot \begin{pmatrix} p_1 \\ p_2 \\ p_3 \end{pmatrix} = \begin{pmatrix} p_1 \\ \cos\alpha \cdot p_2 - \sin\alpha \cdot p_3 \\ \sin\alpha \cdot p_2 + \cos\alpha \cdot p_3 \end{pmatrix}$$

$$\begin{pmatrix} p_1' \\ p_2' \\ p_3' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} p_1 \\ p_2 \\ p_3 \\ 1 \end{pmatrix} = \begin{pmatrix} p_1 \\ \cos\alpha \cdot p_2 - \sin\alpha \cdot p_3 \\ \sin\alpha \cdot p_2 + \cos\alpha \cdot p_3 \\ 1 \end{pmatrix}$$

# Shearing Expressed in Homogeneous Coordinates

$$\begin{pmatrix} p_1' \\ p_2' \\ p_3' \end{pmatrix} = \begin{pmatrix} 1 & m & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} p_1 \\ p_2 \\ p_3 \end{pmatrix} = \begin{pmatrix} p_1 + m \cdot p_2 \\ p_2 \\ p_3 \end{pmatrix}$$




$$\begin{pmatrix} p_1' \\ p_2' \\ p_3' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & m & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} p_1 \\ p_2 \\ p_3 \\ 1 \end{pmatrix} = \begin{pmatrix} p_1 + m \cdot p_2 \\ p_2 \\ p_3 \\ 1 \end{pmatrix}$$

# Shearing: General Case

$$\begin{pmatrix} p_1' \\ p_2' \\ p_3' \end{pmatrix} = \begin{pmatrix} 1 & m_{12} & m_{13} \\ m_{21} & 1 & m_{23} \\ m_{31} & m_{32} & 1 \end{pmatrix} \cdot \begin{pmatrix} p_1 \\ p_2 \\ p_3 \end{pmatrix} = \cdots$$

$$\begin{pmatrix} p_1' \\ p_2' \\ p_3' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & m_{12} & m_{13} & 0 \\ m_{21} & 1 & m_{23} & 0 \\ m_{31} & m_{32} & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} p_1 \\ p_2 \\ p_3 \\ 1 \end{pmatrix} = \begin{pmatrix} p_1 + m_{12} \cdot p_2 + m_{13} \cdot p_3 \\ p_2 + m_{21} \cdot p_1 + m_{23} \cdot p_3 \\ p_3 + m_{31} \cdot p_1 + m_{32} \cdot p_2 \\ 1 \end{pmatrix}$$

# Computational Complexity for 3D Transformations

$$\begin{pmatrix} p'_1 \\ p'_2 \\ p'_3 \\ 1 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & t_1 \\ a_{21} & a_{22} & a_{23} & t_2 \\ a_{31} & a_{32} & a_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} p_1 \\ p_2 \\ p_3 \\ 1 \end{pmatrix} = \begin{pmatrix} a_{11} \cdot p_1 + a_{12} \cdot p_2 + a_{13} \cdot p_3 + t_1 \\ a_{21} \cdot p_1 + a_{22} \cdot p_2 + a_{23} \cdot p_3 + t_2 \\ a_{31} \cdot p_1 + a_{32} \cdot p_2 + a_{33} \cdot p_3 + t_3 \\ 1 \end{pmatrix}$$

- Operations needed:
  - 9 multiplications
  - 9 additions
- … for an arbitrarily complex affine 3D transformation (of a position vector)
- Runtime complexity improved by pre-calculation of composed transformation matrices
  - Hardware implementations in graphics processors
  - Very efficient

# Chapter 2 – Transformations & Scene Graphs

- Three-Dimensional Geometric Transformations
- Affine Transformations and Homogeneous Coordinates
- Combining Transformations

- Why a scene graph?
- What is stored in the scene graph?
  - Objects
  - Appearance
  - Camera
  - Lights
- Rendering with a scene graph
- Practical example

# Combining several transformations: Order matters!

$$p' = A \cdot B \cdot p = A \cdot (B \cdot p) = (A \cdot B) \cdot p \boxed{\neq (B \cdot A) \cdot p}$$

$$p = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix} \quad A = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad B = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 5 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

A = Rotation 90° around X axis (i.e., Y becomes Z)
B = Translation by 5 along Y axis

ABp = A(Bp) means: first translate, then rotate the result
BAp = B(Ap) means: first rotate, then translate the result

$$(A \cdot B) \cdot p = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 5 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 5 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 5 \\ 1 \end{pmatrix}$$

$$(B \cdot A) \cdot p = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 5 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 5 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 5 \\ 0 \\ 1 \end{pmatrix}$$

# The same example in Three.js

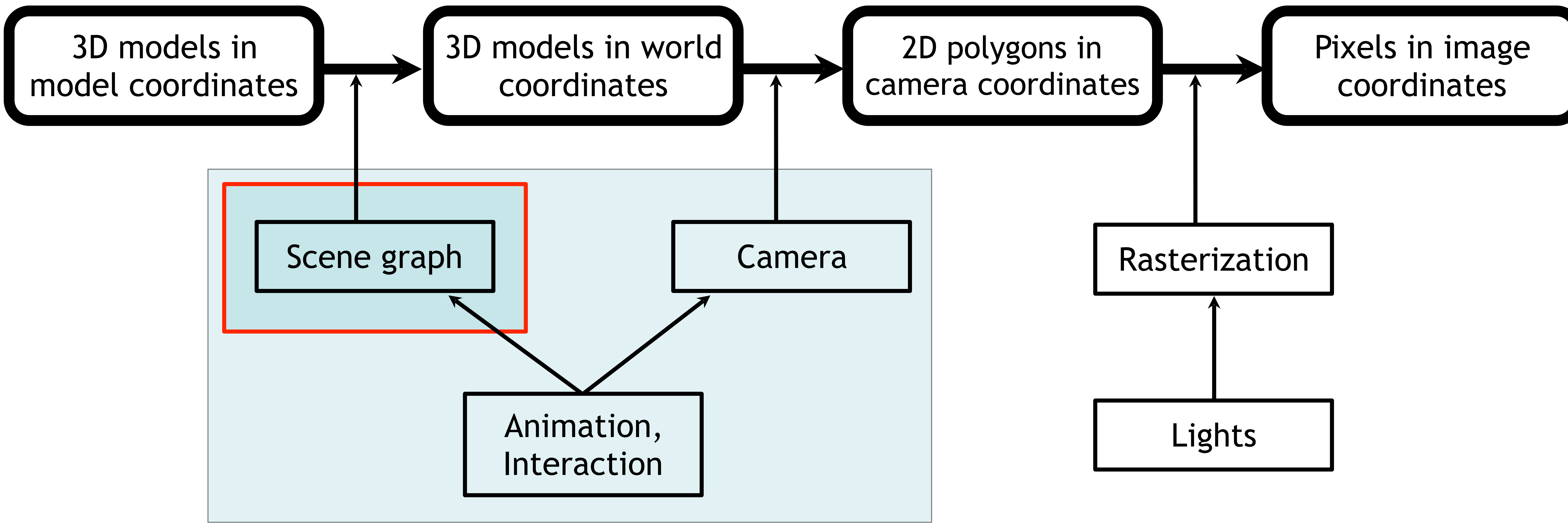
```
var p = new THREE.Vector4( 1, 0, 0, 1);

var M = new THREE.Matrix4(); // initialized by identity

var A = new THREE.Matrix4();
var B = new THREE.Matrix4();

var gamma = Math.PI / 2; // equals 90 degrees

A.makeRotationX( gamma ); // rotation by 90 degrees around X axis
B.makeTranslation( 0, 5, 0 ); // translation by 5 along Y axis

M.multiply( A ); // Now M contains MA = A
M.multiply( B ); // Now M contains AB

p.applyMatrix4( M ); // Now p contains ABp
```

# Chapter 2 – Transformations & Scene Graphs

- Three-Dimensional Geometric Transformations
- Affine Transformations and Homogeneous Coordinates
- Combining Transformations

- Why a scene graph?
- What is stored in the scene graph?
  - Objects
  - Appearance
  - Camera
  - Lights
- Rendering with a scene graph
- Practical example

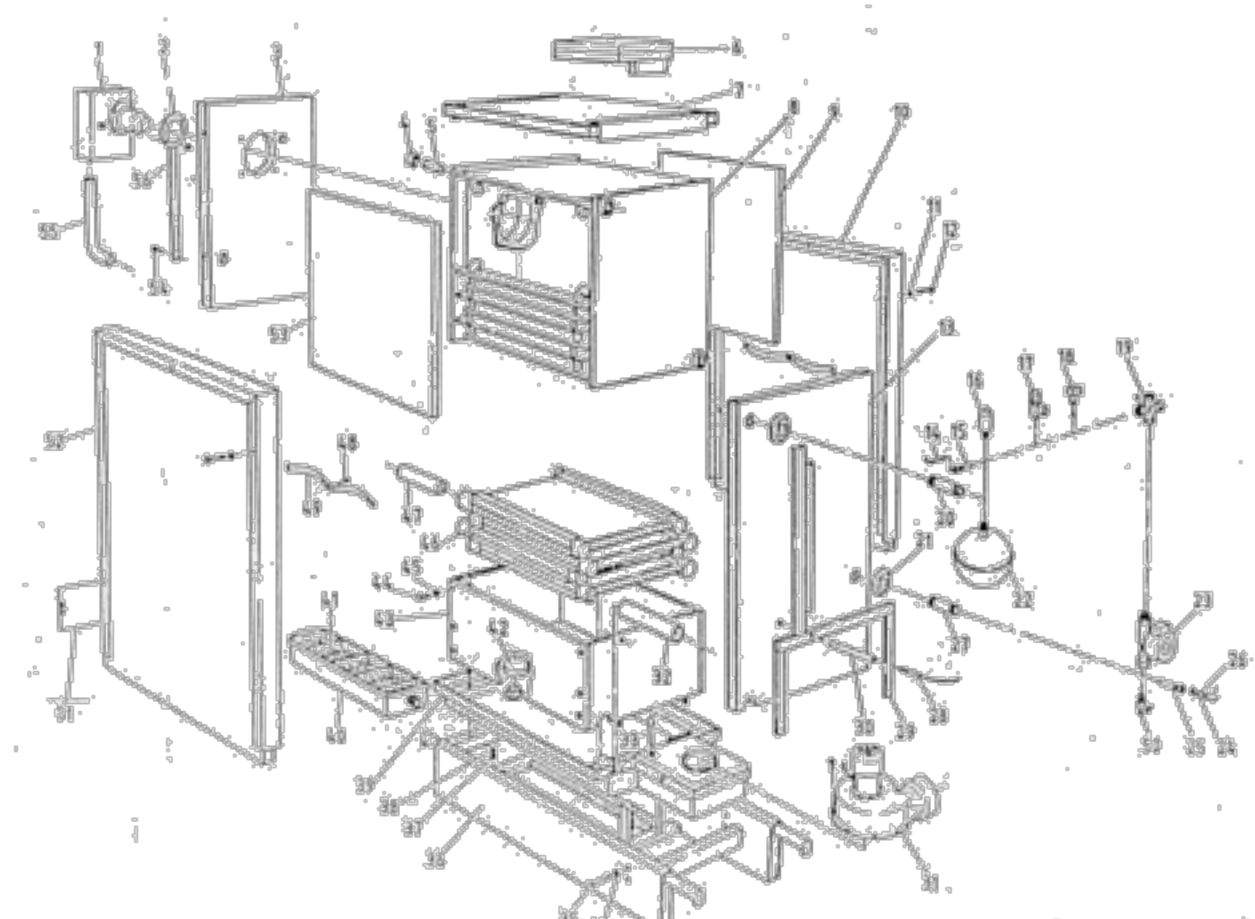
# The 3D Rendering Pipeline (our version for this class)



# Why a Scene Graph?

- Naive approach:
  - For each object in the scene, set its transformation by a single matrix (i.e., a tree 1 level deep and N nodes wide)
    - Advantage: very fast for rendering
    - Disadvantage: if several objects move in the same way, all of their transforms change
- Observation: Things in the world are made from parts
- Approach: define an object hierarchy along the *part-of* relation
  - Transform all parts only relative to the whole group
  - Transform group as a whole with another transform
  - Parts can be groups again

https://www.watersafetyshop.com/media/apeks/drawings/apeks-1-stage-flight-800px.jpg



http://www.bosy-online.de/Veritherm/Explosionszeichnung.jpg

# Chapter 2 – Transformations & Scene Graphs

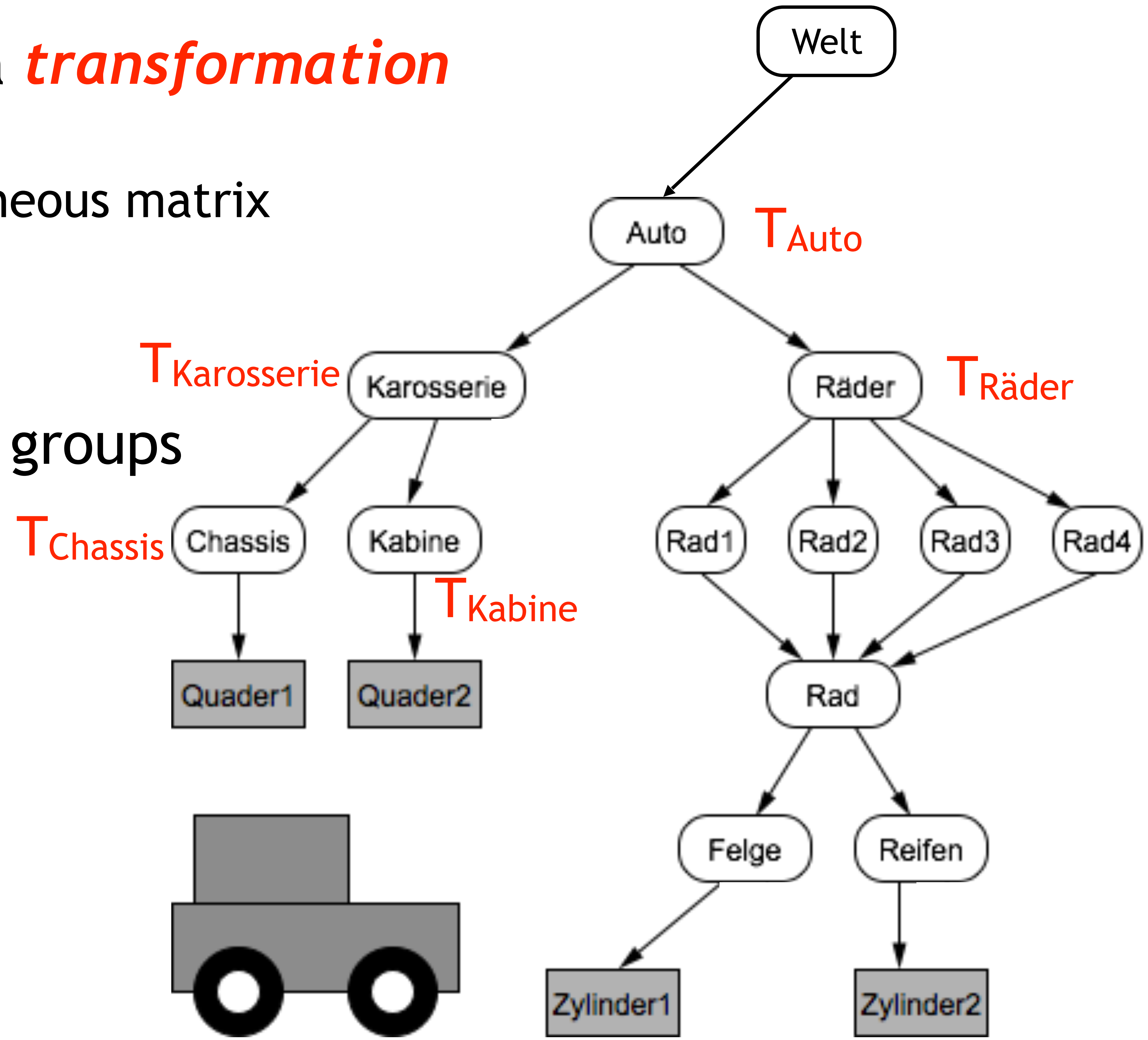
- Three-Dimensional Geometric Transformations
- Affine Transformations and Homogeneous Coordinates
- Combining Transformations

- Why a scene graph?
- What is stored in the scene graph?
  - Objects
  - Appearance
  - Camera
  - Lights
- Rendering with a scene graph
- Practical example

# Geometry in the Scene Graph

- Leaves are basic 3D objects
- Non-leaf nodes (groups) contain a ***transformation***
  - can have one or several children
  - transformation is given by a homogeneous matrix
- Root is the entire world

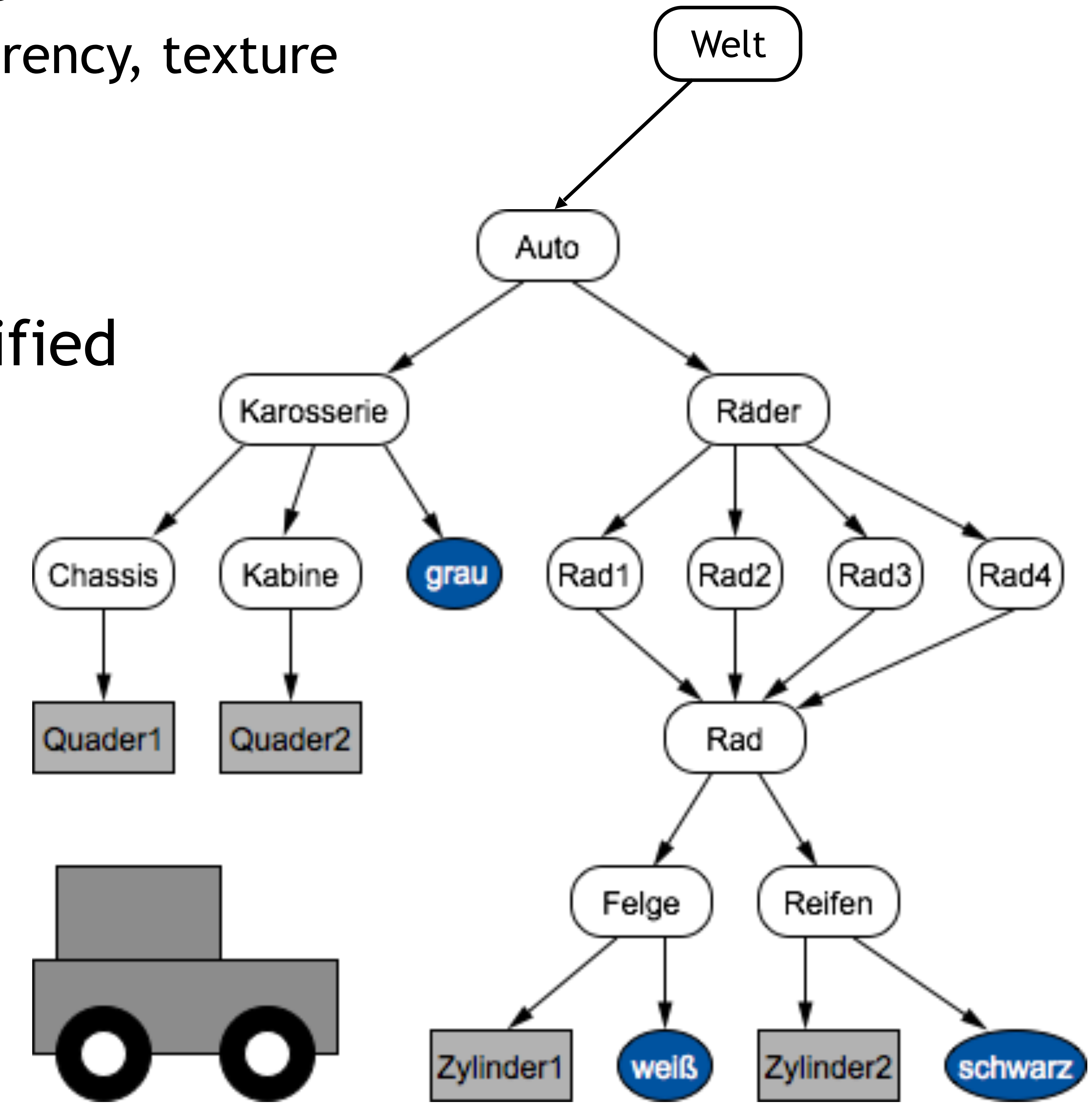
- Nodes can be the child of several groups
  - Not a tree, but a directed acyclic graph (DAG)
  - Effective reuse of geometry



# Appearance in the Scene Graph

- Scene graph also contains appearances
  - Appearance: e.g. Color, reflection, transparency, texture
    → Details see next lecture(s)
  - Can be reused similarly to geometry

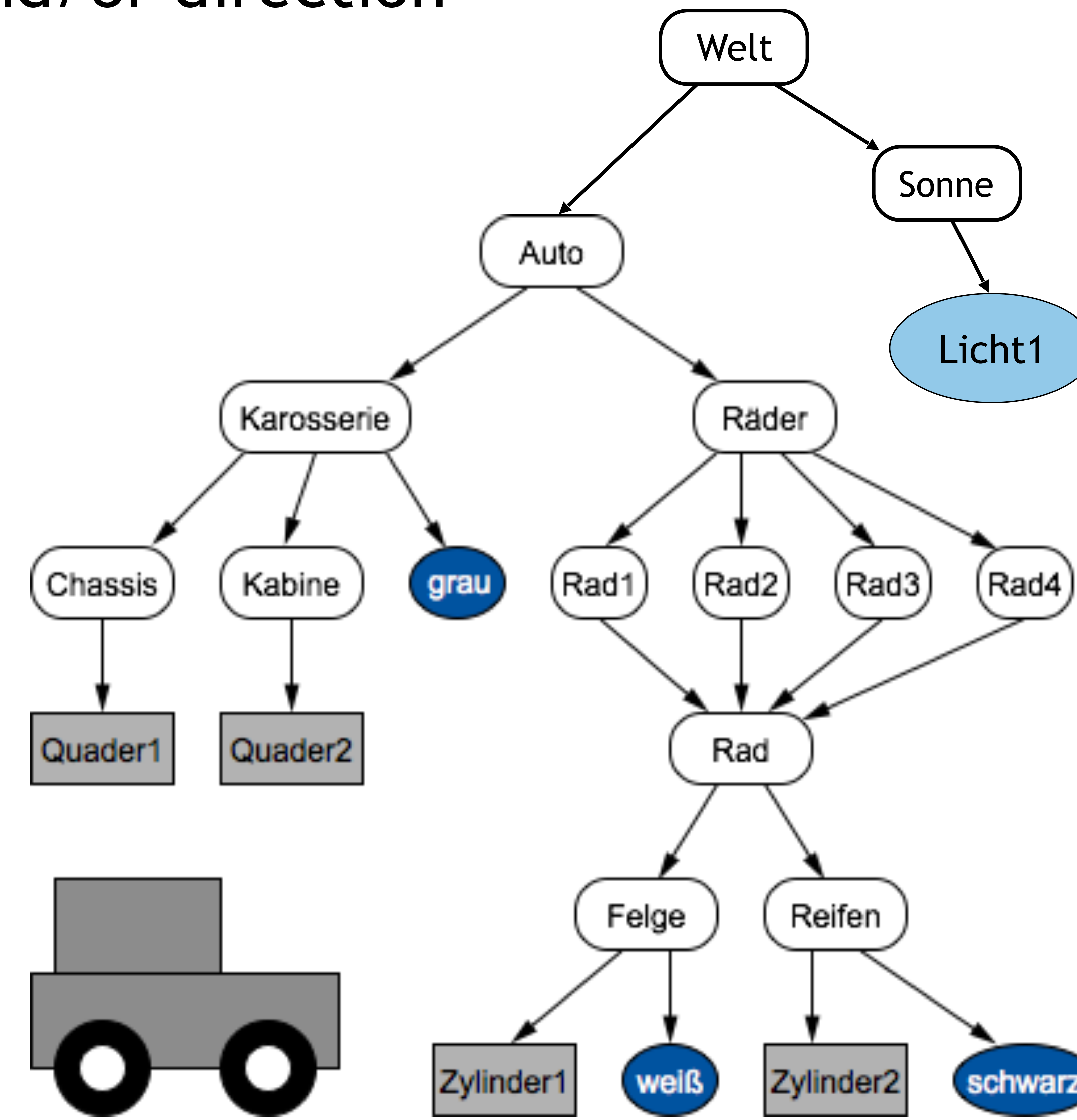
- Appearance can be only partially specified
  - Unspecified values are inherited



# Lights in the Scene Graph

- Light sources also need a position and/or direction
  - Just include them into the scene graph
  - Can be animated just like geometry

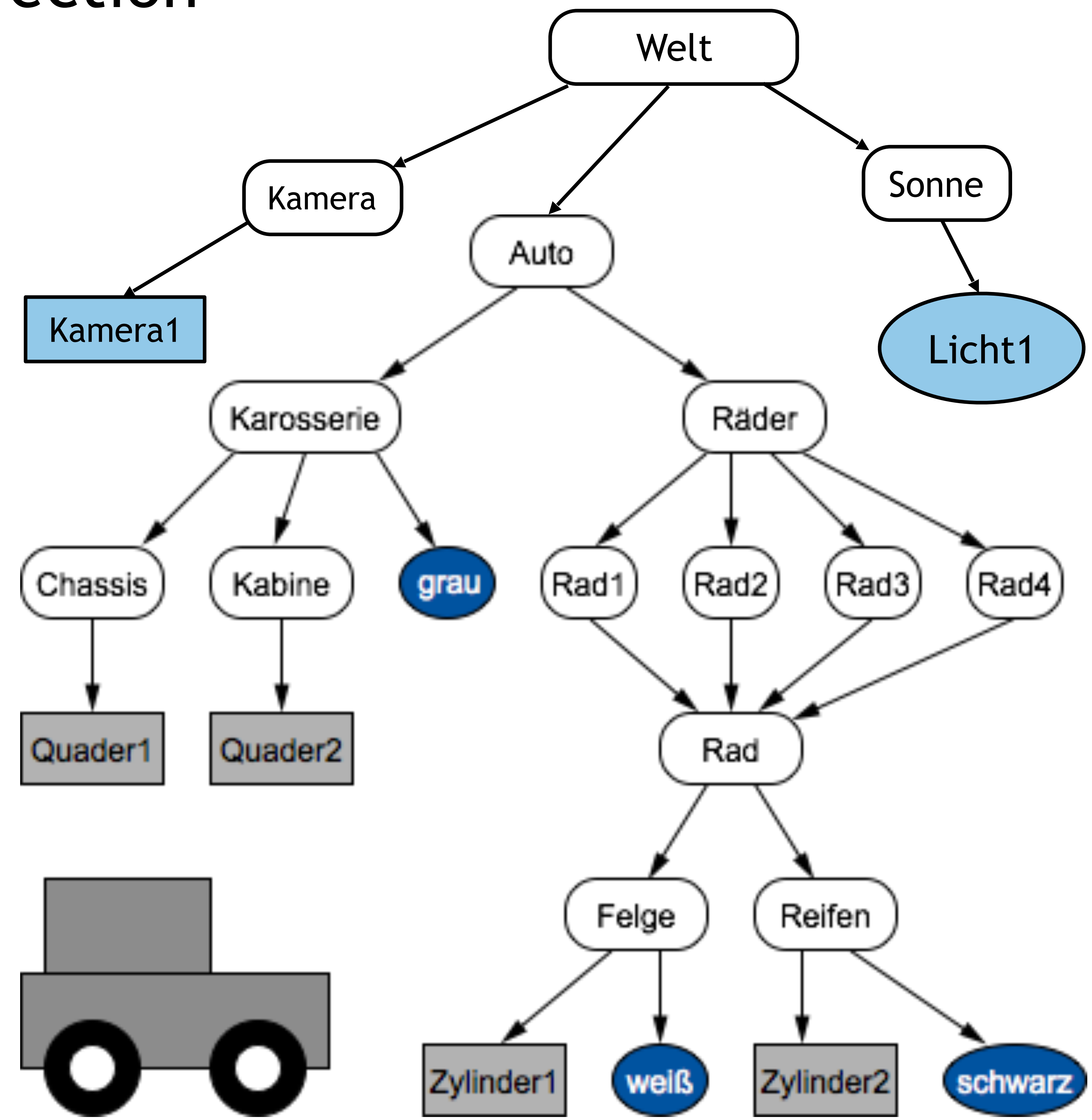
- Lights can be in local coordinate systems of geometry groups
  - Move with them
  - Example: lights on a car



# The Camera in the Scene Graph

- Camera also needs a position and direction
  - Just include it into the scene graph
  - Can be animated just like geometry

- Camera can be in local coordinate systems of geometry groups
  - Move with them
  - Example: driver's view from a car



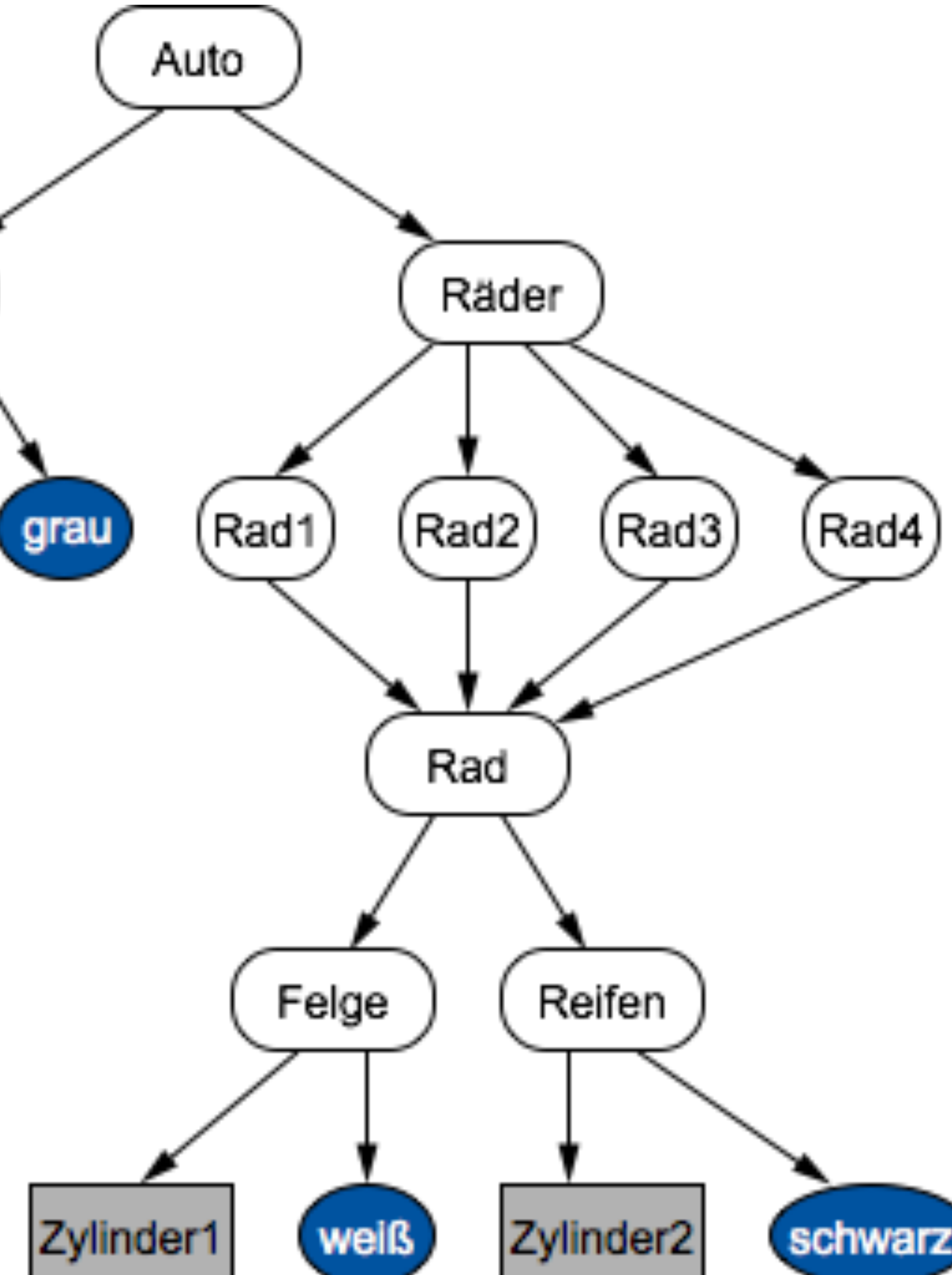
# Chapter 2 – Transformations & Scene Graphs

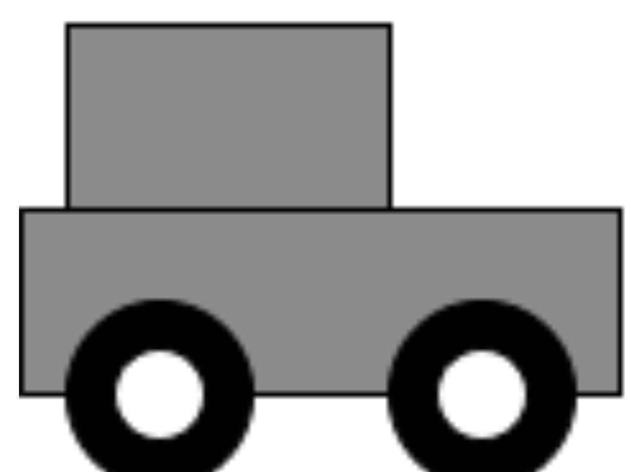
- Three-Dimensional Geometric Transformations
- Affine Transformations and Homogeneous Coordinates
- Combining Transformations

- Why a scene graph?
- What is stored in the scene graph?
  - Objects
  - Appearance
  - Camera
  - Lights
- Rendering with a scene graph
- Practical example

# Scene graph traversal for rendering

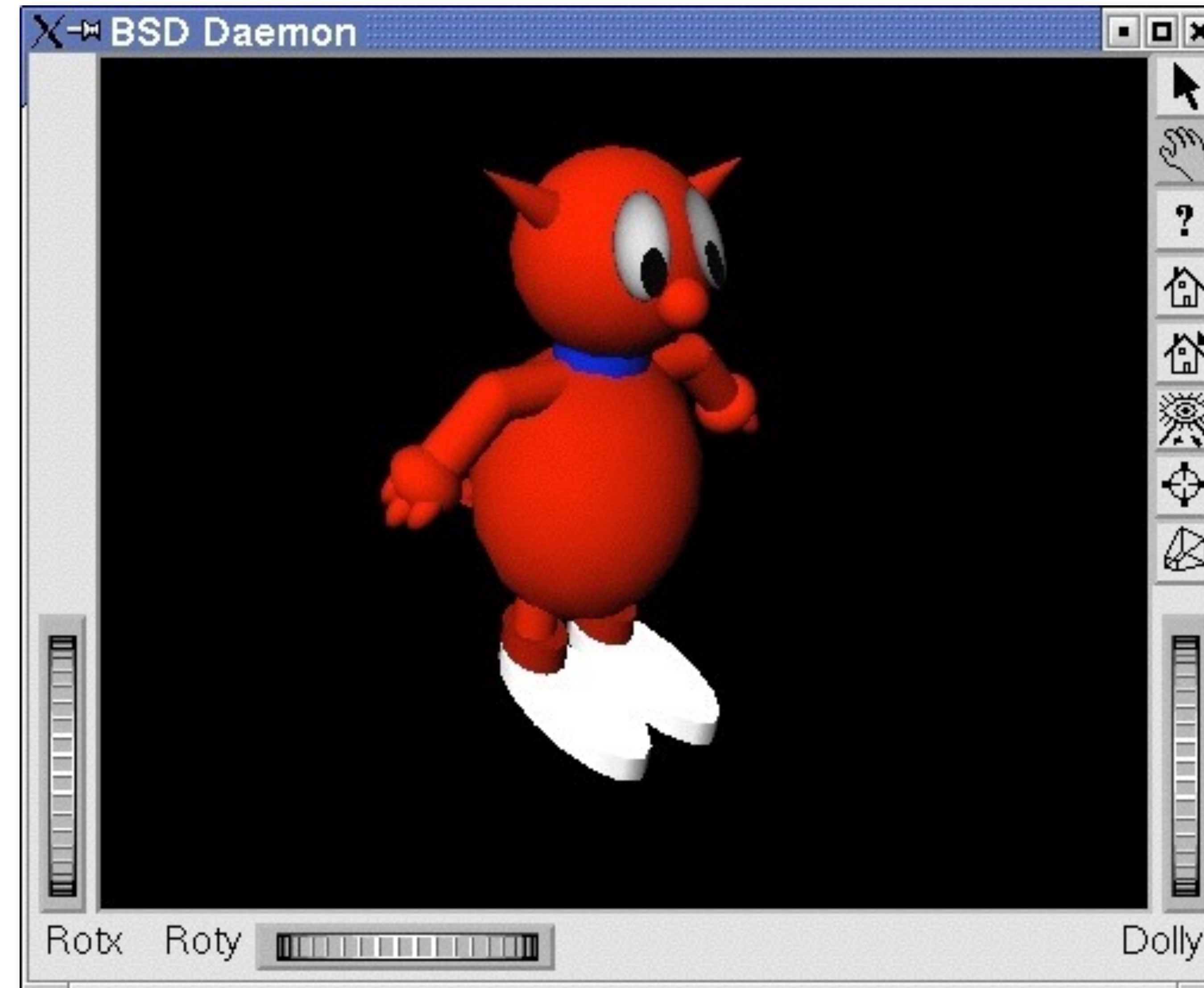
- set $T_{act}$ to $T_{Auto}$
- push state
- set $T_{act}$ to $T_{act} * T_{Karosserie}$
- push state
- set $T_{act}$ to $T_{act} * T_{Chassis}$
- render Quader1
- pop state
- set $T_{act}$ to $T_{act} * T_{Kabine}$
- render Quader2
- pop state
- pop state
- set $T_{act}$ to $T_{act} * T_{Räder}$
- …



# Scene Graph Libraries

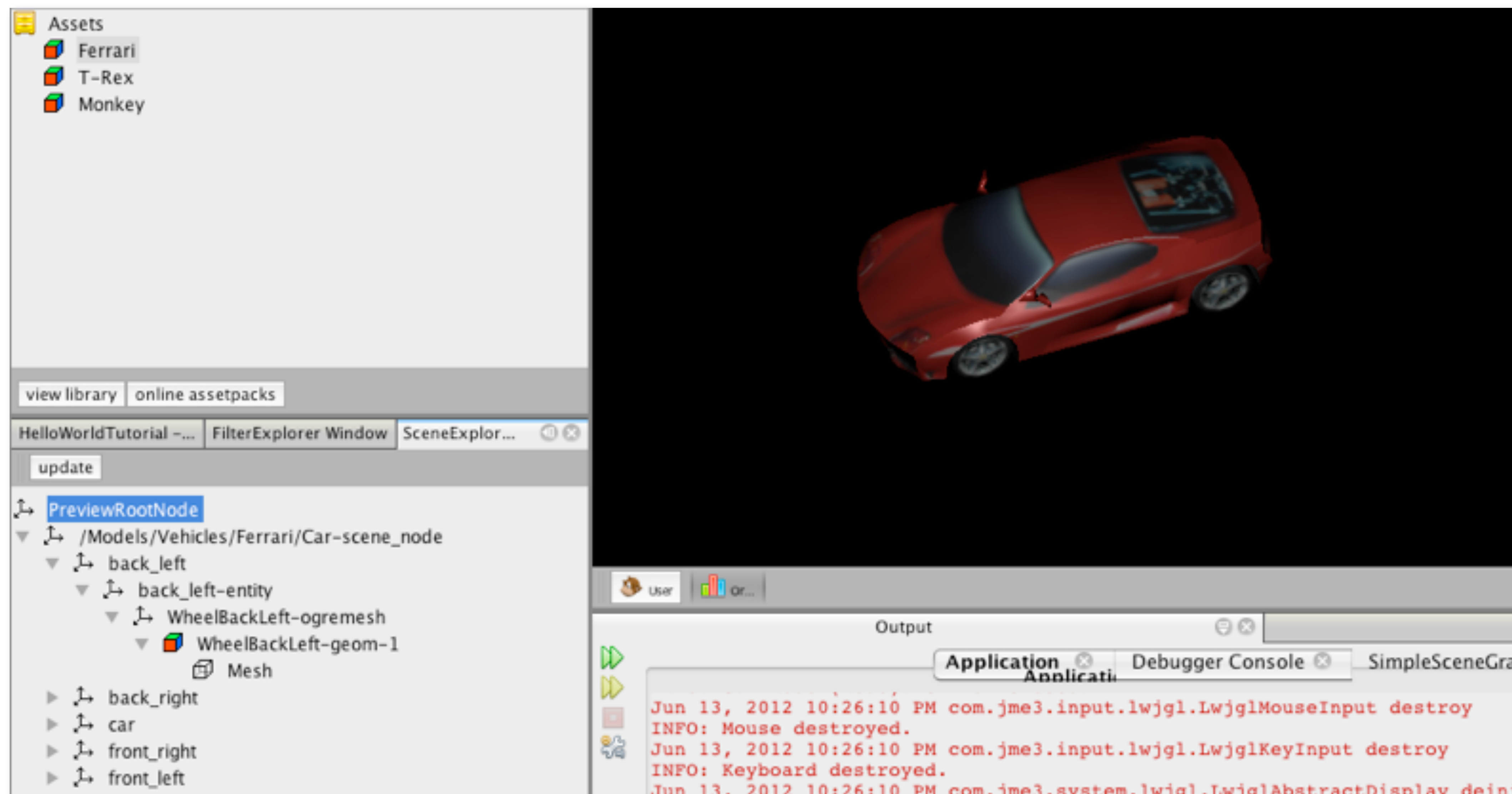
- Scene graphs exist on a more abstract layer than OpenGL!
- VRML/X3D
  - descriptive text format, ISO standard
- OpenInventor
  - Based on C++ and OpenGL
  - Originally Silicon Graphics, 1988
  - Now supported by VSG3d.com
- Java3D
  - Provides 3D data structures in Java
  - *Not supported anymore*
- Open Scene Graph (OSG)
- Various game engines
  - e.g. Unity or jMonkey Engine (scene graph based game engine for Java)
- Three.js




http://www.shlomifish.org/open-source/bits-and-bobs/open-inventor-bsd-daemon/

# Scene Graphs in Practice

- Creation of scene graphs and objects
  - Specific authoring software (e.g. Blender, Maya, 3DS Max)
- Assets (models, objects) exported to exchange formats
  - E.g. (X3D,) Wavefront OBJ (.obj), 3ds Max (.3ds), Ogre XML (.mesh)
- Objects typically are tesselated
  - Polygon meshes
  - No primitive geometric objects visible/readable anymore
- Example:
  - jMonkey Engine Scene Viewer / Composer



# Chapter 2 – Transformations & Scene Graphs

- Three-Dimensional Geometric Transformations
- Affine Transformations and Homogeneous Coordinates
- Combining Transformations

- Why a scene graph?
- What is stored in the scene graph?
  - Objects
  - Appearance
  - Camera
  - Lights
- Rendering with a scene graph
- Practical example

# Example of a scene graph

- Graph to be drawn together in the lecture
- VRML world linked from the class page