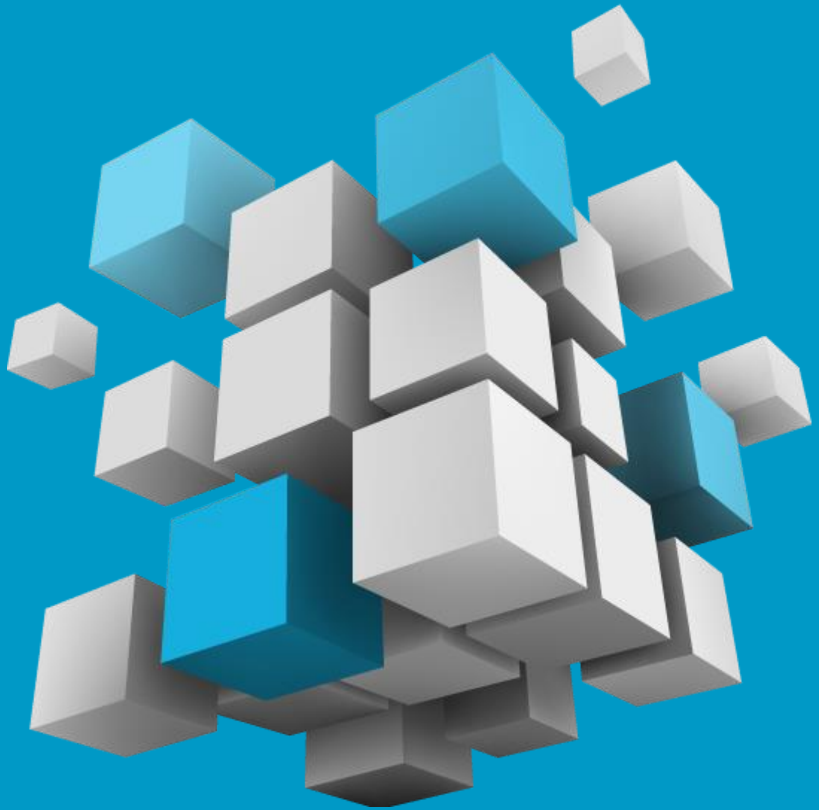
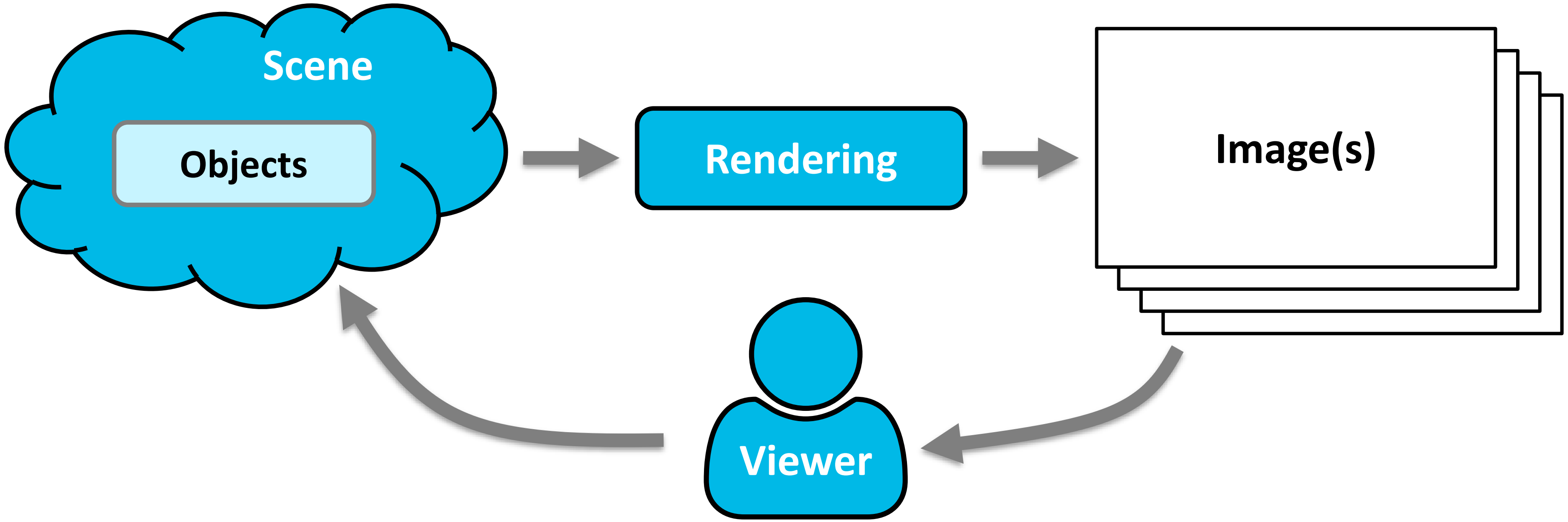
Introduction to Computer Graphics

Hierarchical transformations

Scene graphs



Computer Graphics Pipeline



Scene contains list of objects

- Iterate over objects to render images
- Each object with its own transformation

3D Transformations

Expressed as 4x4 matrices

- Affine transformation in homogeneous coordinates
- 4th column: translation
- 4th row: perspective projection

Transformation can be combined

- Matrix multiplication (from left)

Transform $\boldsymbol{p} = (x\ y\ z)^{\mathrm{T}}$ by $\mathbf{M}$

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \mathbf{M} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \qquad \mathbf{M} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

3D Transformations

Translation

$$\mathbf{T} = \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

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

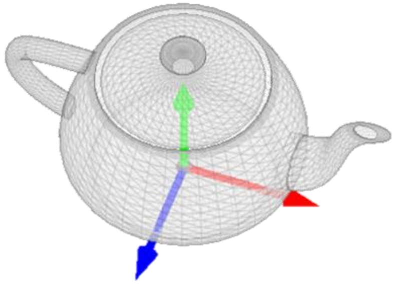
$$\mathbf{R}_y = \begin{pmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

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

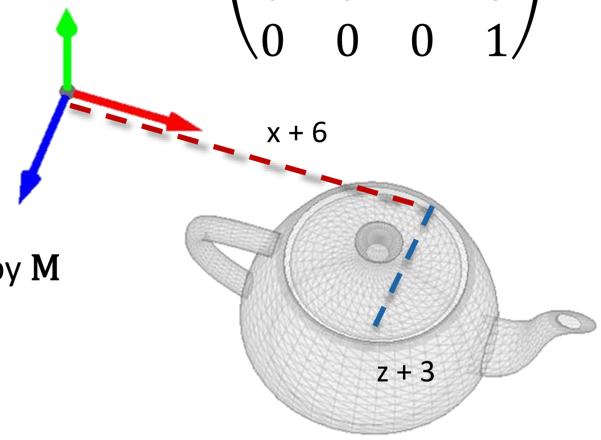
Rotations around x, y, z axis

Scaling

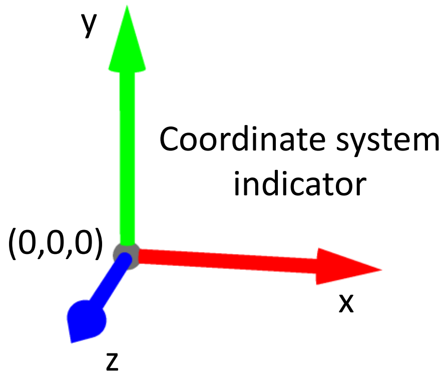
$$\mathbf{S} = \begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



$$\mathbf{M} = \begin{pmatrix} 1 & 0 & 0 & 6 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 3 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



Transform $\boldsymbol{p} = (\boldsymbol{x}\ \boldsymbol{y}\ \boldsymbol{z})^{\mathrm{T}}$ by $\mathbf{M}$



Euler Rotation

General rotation built from 3 subsequent rotations

- Rotation axes are orthogonal (XYZ)
- Euler angles describe the three rotation angles

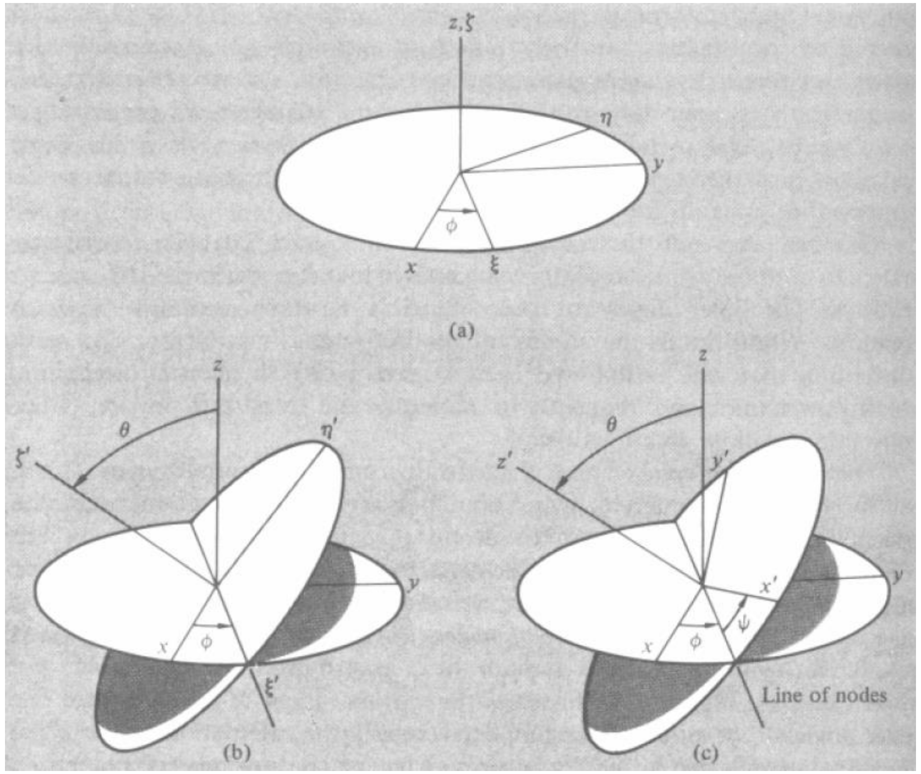
12 possible rotation sequences

Often used definition of Euler angles: z-x-z

- Passive representation
  1. Rotation with ϕ around z
  2. Rotation with θ around ξ (current x axis)
  3. Rotation with ψ around ζ' (current z axis)
- As rotation matrix (4x4)

$$\mathbf{R} = \mathbf{R}_{z,\Psi}\mathbf{R}_{x,\theta}\mathbf{R}_{z,\phi}$$

Cannot distinguish ϕ and ψ for small rotations



z-x-z Euler rotation

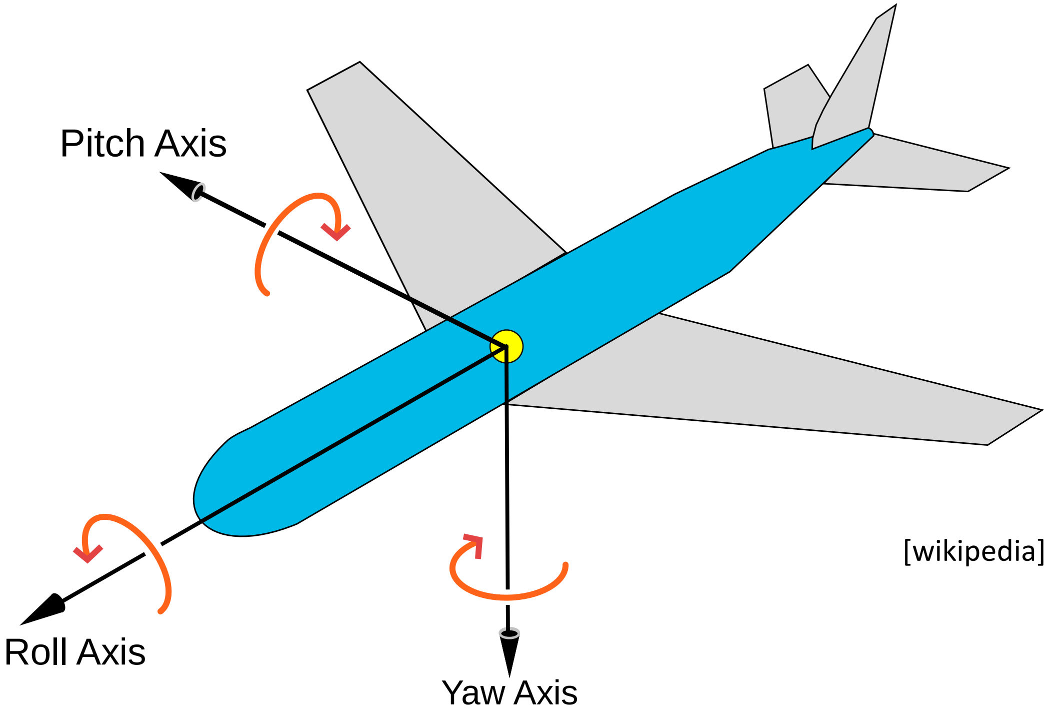
Euler Rotation (cont.)

In animation & engineering (aircraft and satellite control)

- x-y-z convention (Tait-Bryan angles)

Active representation

- Rotation around 3 stationary axes
- Around x: roll
- Around y: pitch/altitude
- Around z: yaw/heading

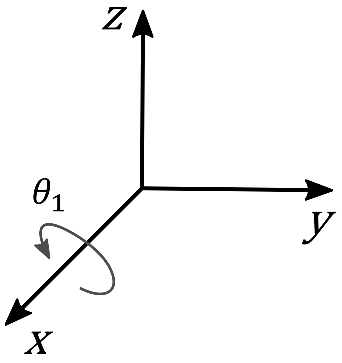


[wikipedia]

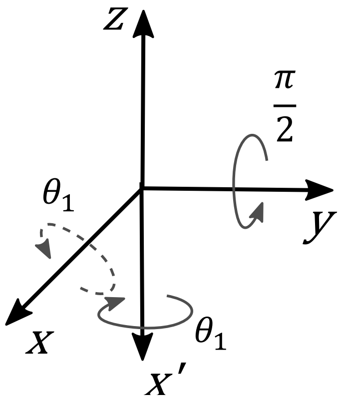
Euler Rotation Problems (cont.)

Gimbal lock

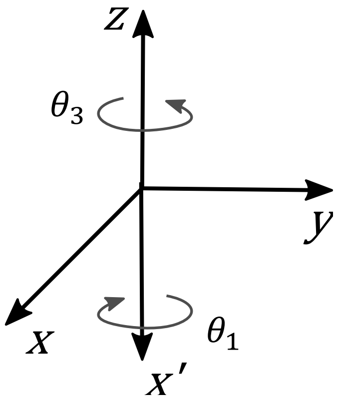
- One rotation axis falls onto another
- Loss of one degree of freedom



x rotation θ_1



x rotation θ_1,
followed by y rotation $\pi/2$

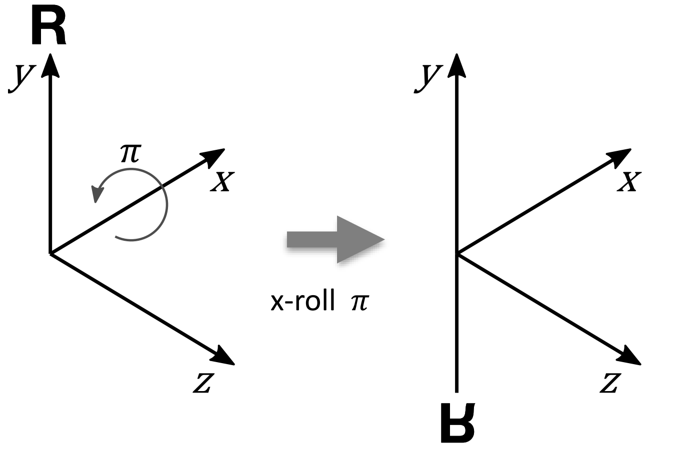


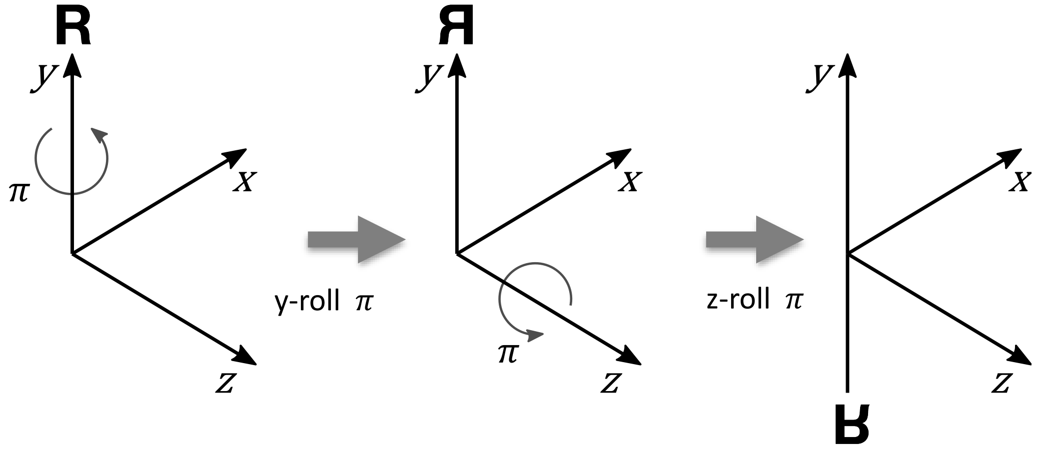
z rotation θ_3,
identical to x rotation $-\theta_3$

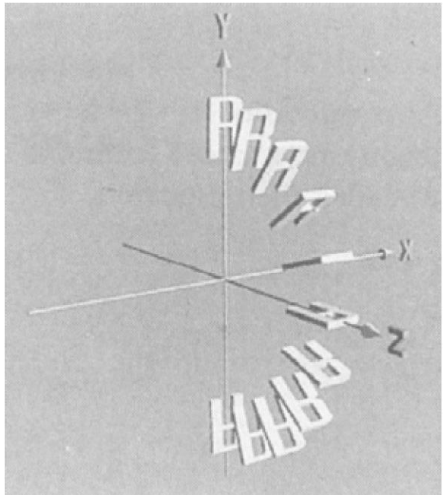
➔**Solution:** Quaternions

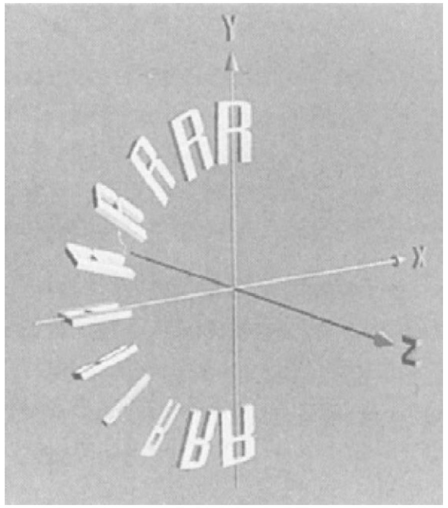
Euler Rotation Problems (cont.)

Euler angles are ambiguous









Rotation Quaternions

Compact representation of rotations

Based on complex numbers $\boldsymbol{q} = s + x\mathrm{i} + y\mathrm{j} + z\mathrm{k} \quad s, x, y, z \in \mathbb{R}$

- Not a vector with 4 components!

Quaternion **q** for rotation around axis $\boldsymbol{v}$ by angle θ

$$\boldsymbol{q} = \left[\cos\frac{1}{2}\theta\,, \sin\frac{1}{2}\theta\,\boldsymbol{v}\right] \qquad \theta \in [0, \pi]$$

Normalized rotation axis $(x\ y\ z)$

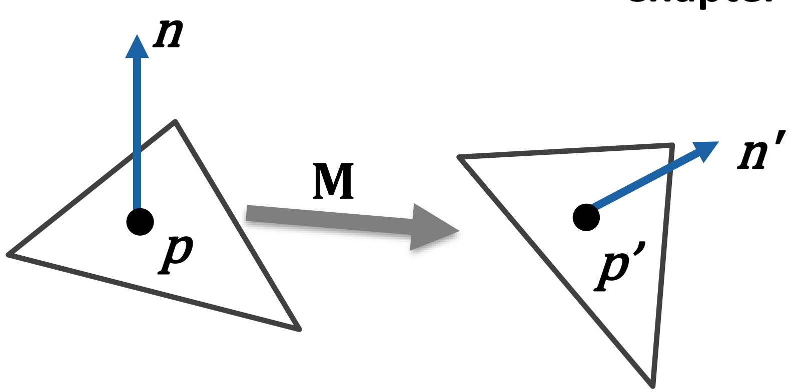
Rotate point **p** with **q**

$$\boldsymbol{p}' = \boldsymbol{q}\boldsymbol{p}\boldsymbol{q}^{-1}$$

Used for interpolation between points on a sphere

- Spherical linear interpolation (SLERP)

Transforming the Normal Vector



Cannot use **M** to transform normals in general case due to shearing, non-uniform scaling, …

$$\boldsymbol{p} = \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \text{ is on plane } E\text{: } ax + by + cz + d = 0$$

$$\text{Normal } \boldsymbol{n} = \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix}$$

$$E\text{:} \quad \underbrace{\begin{pmatrix} a & b & c & d \end{pmatrix}}_{\boldsymbol{n}^{\mathrm{T}}} \overbrace{\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}}^{\boldsymbol{p}} = 0$$

With $\boldsymbol{n}^{\mathrm{T}} \cdot \boldsymbol{p} = 0$ and $\boldsymbol{p}' = \mathbf{M} \cdot \boldsymbol{p}$ follows

$$\boldsymbol{n}^{\mathrm{T}}(\mathbf{M}^{-1} \cdot \mathbf{M})\boldsymbol{p} = 0$$

$$\left(\boldsymbol{n}^{\mathrm{T}} \cdot \mathbf{M}^{-1}\right) \cdot (\mathbf{M} \cdot \boldsymbol{p}) = \boldsymbol{n}'^{\mathrm{T}} \cdot \boldsymbol{p}' = 0$$

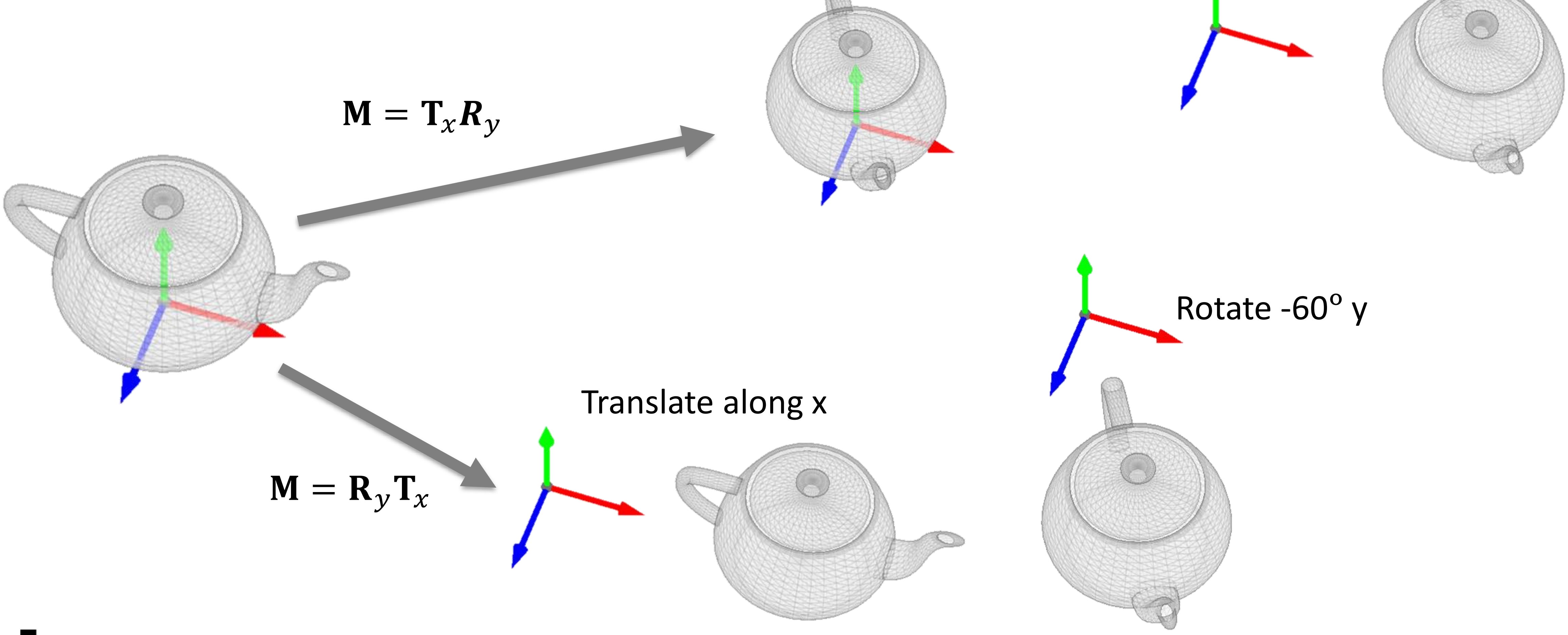
Also known as
Normal Matrix (3x3)

$$\boldsymbol{n}' = \left(\mathbf{M}^{-1}\right)^{\mathbf{T}} \cdot \boldsymbol{n}$$

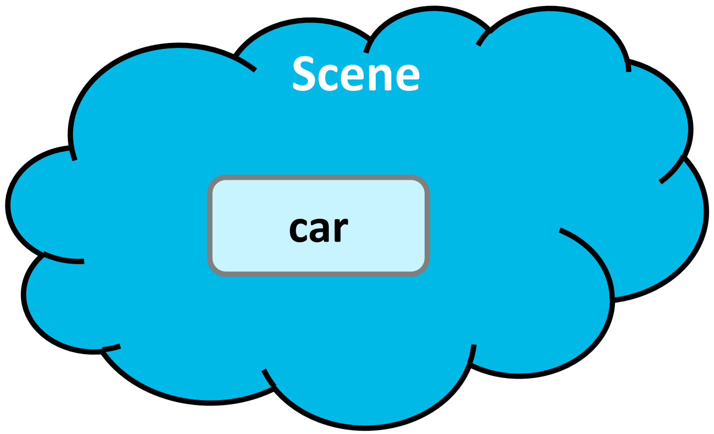
Order of Transformation Matters

How is a point transformed?

- Read transformations from **right** to **left**

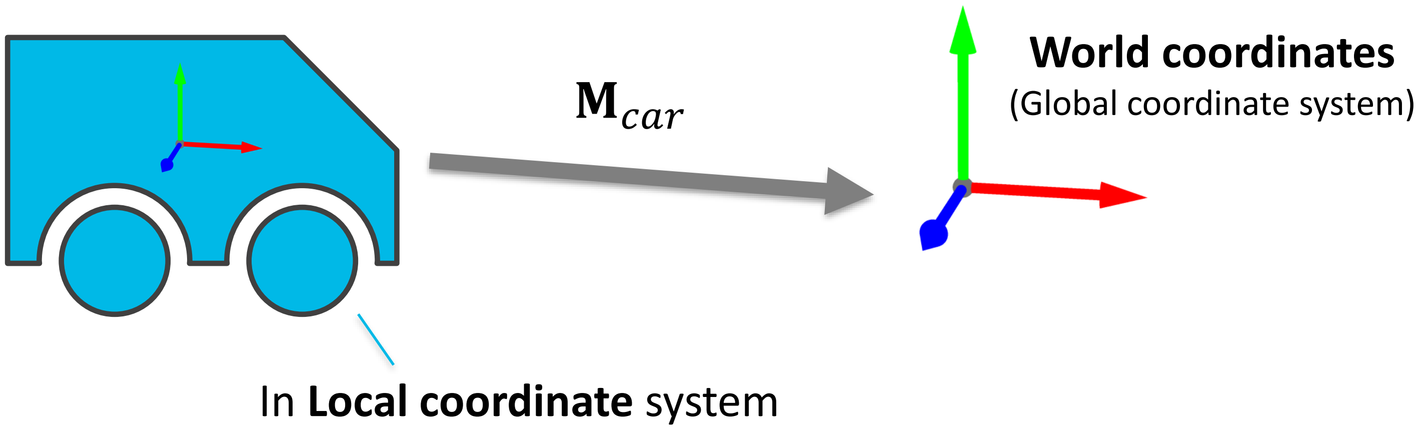


Example: Car

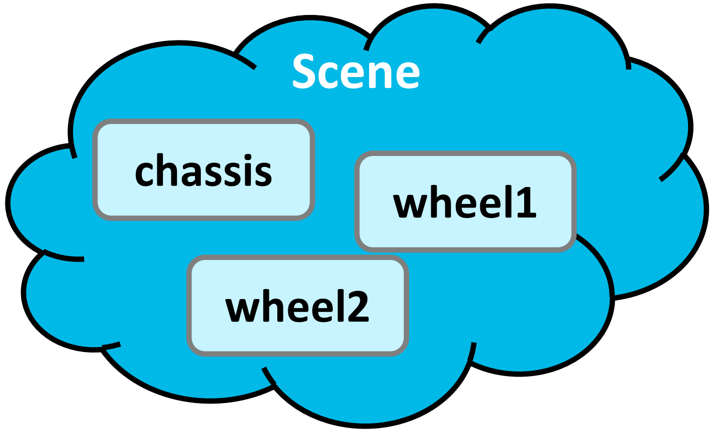


Each object with own transformation **M**

- Transforms everything from **local** to **global** coordinates
- How can we reuse parts? Spin the wheels?

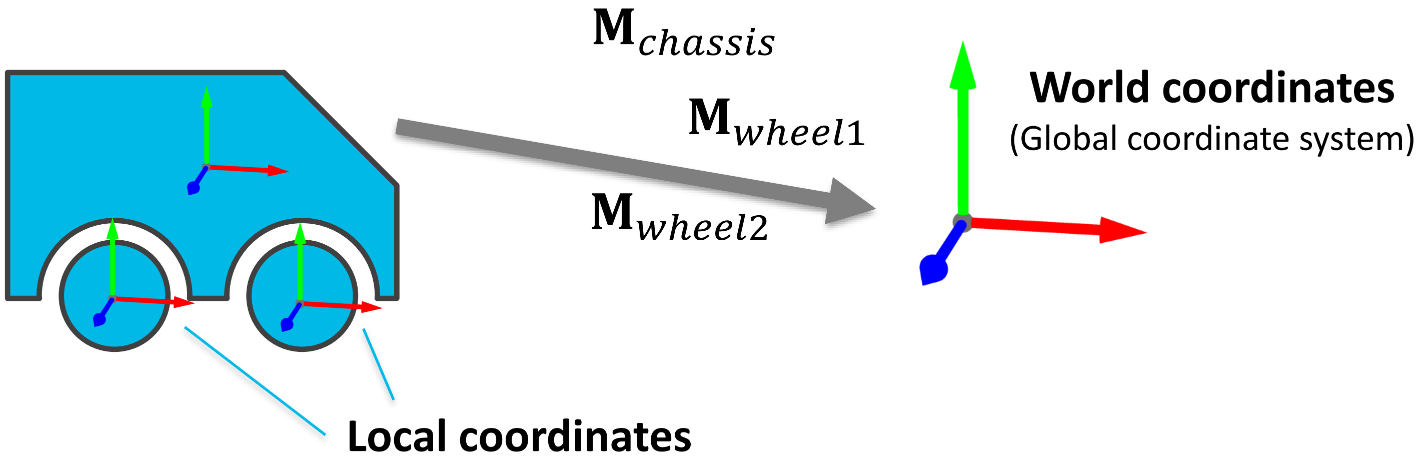


Example: Car Components

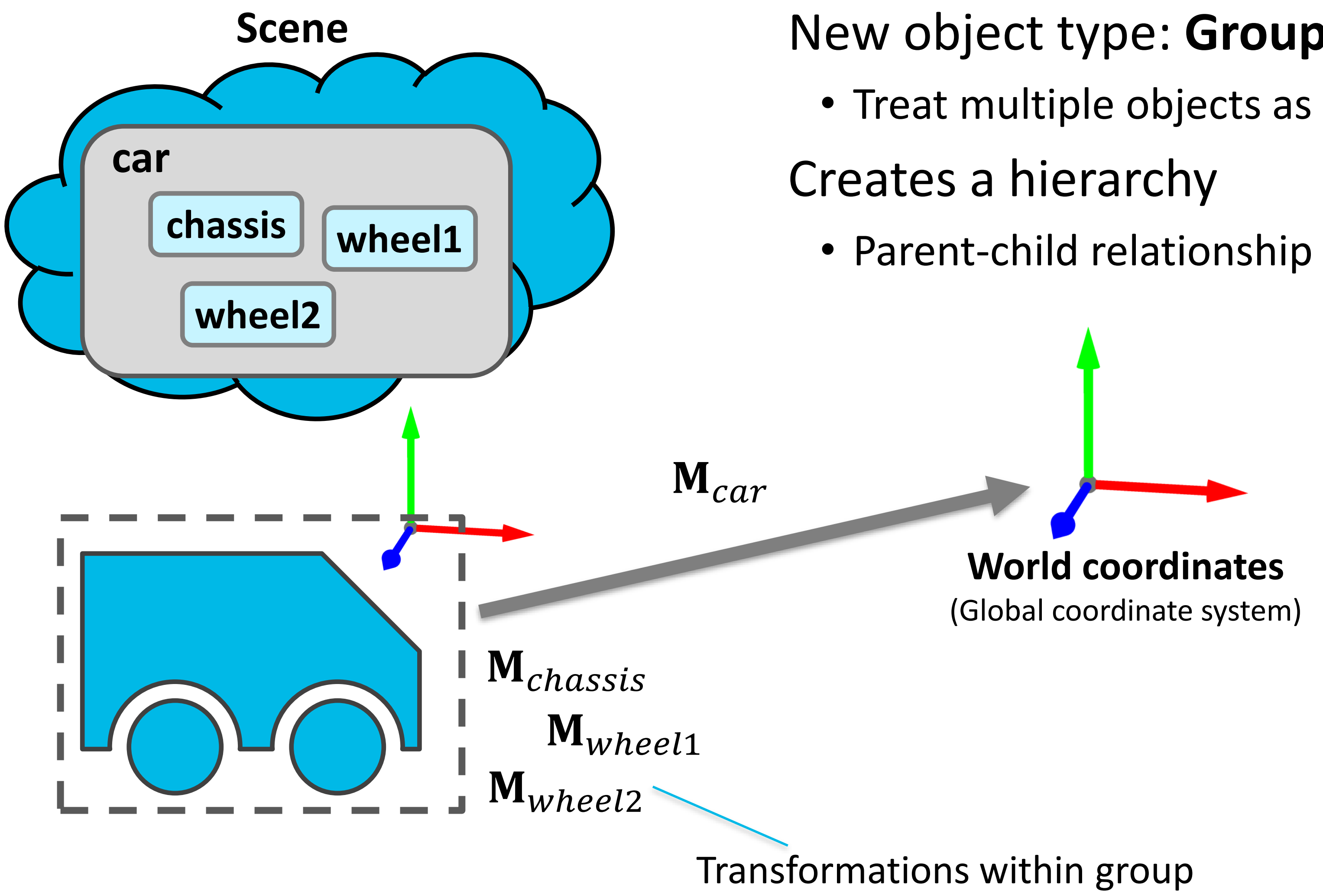


Each object with own transformation **M**

- Transforms everything from **local** to **global** coordinates
- Editing requires updating many transformations
  (translating the car requires 3 updates, rotation even more)



Object Grouping

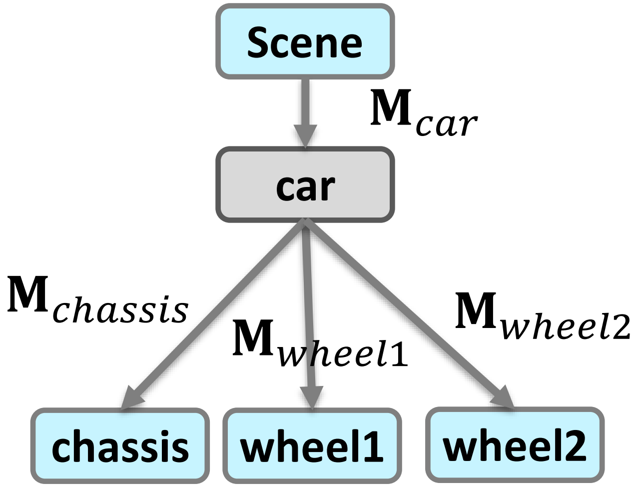


New object type: **Group**

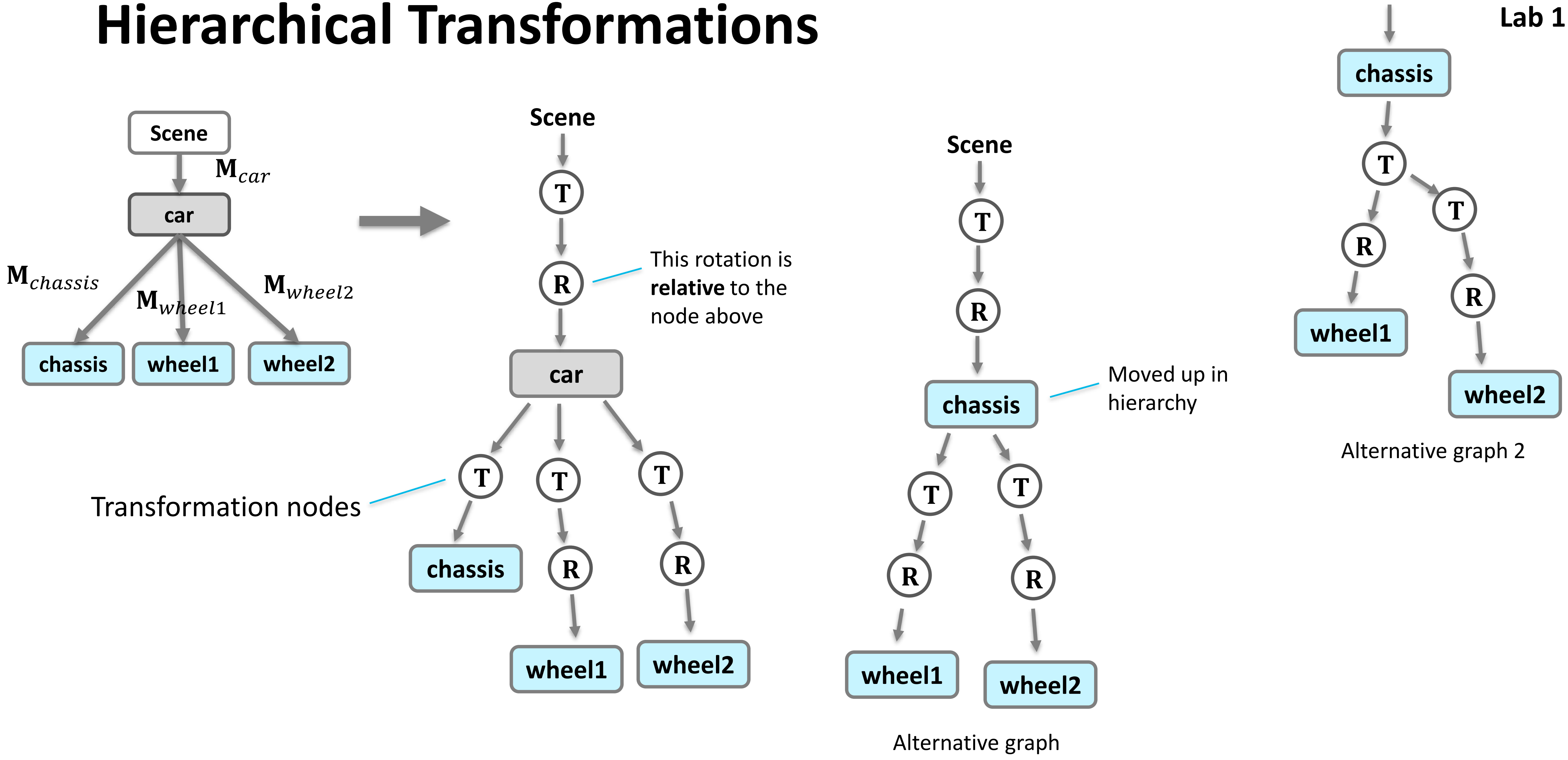
- Treat multiple objects as one

Creates a hierarchy

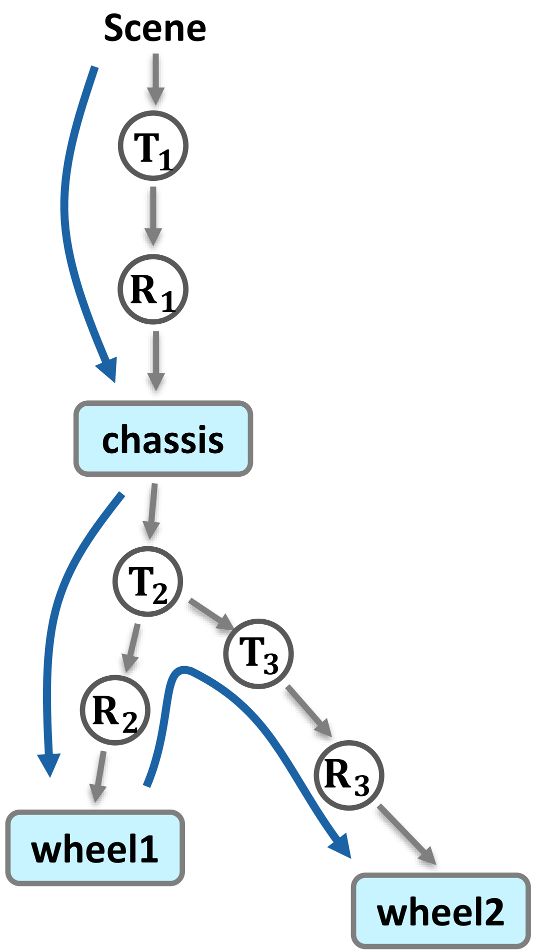
- Parent-child relationship



Hierarchical Transformations



Traversal of Transformations



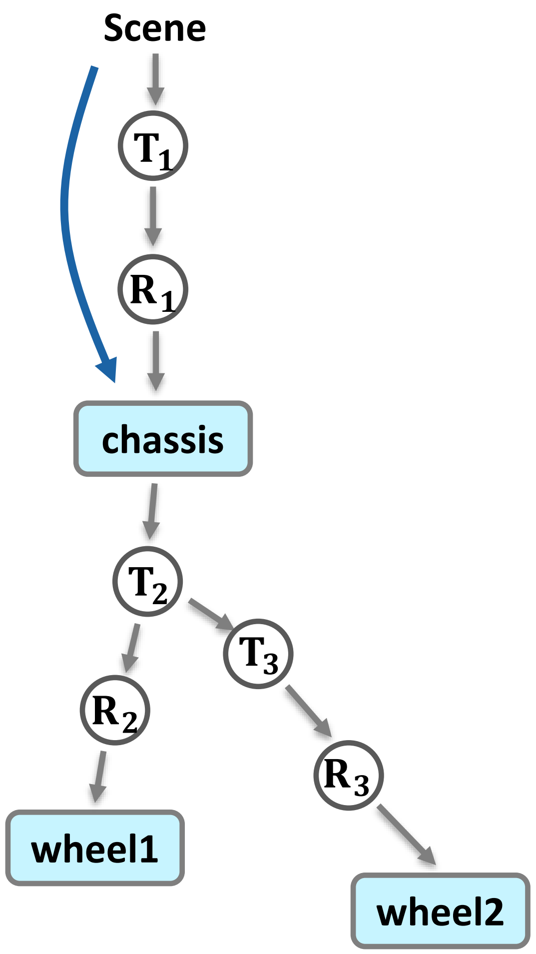
Need to visit all transformations

- Depth first traversal

Accumulate transformations when going down

Undo transformation on way up

Traversal of Transformations

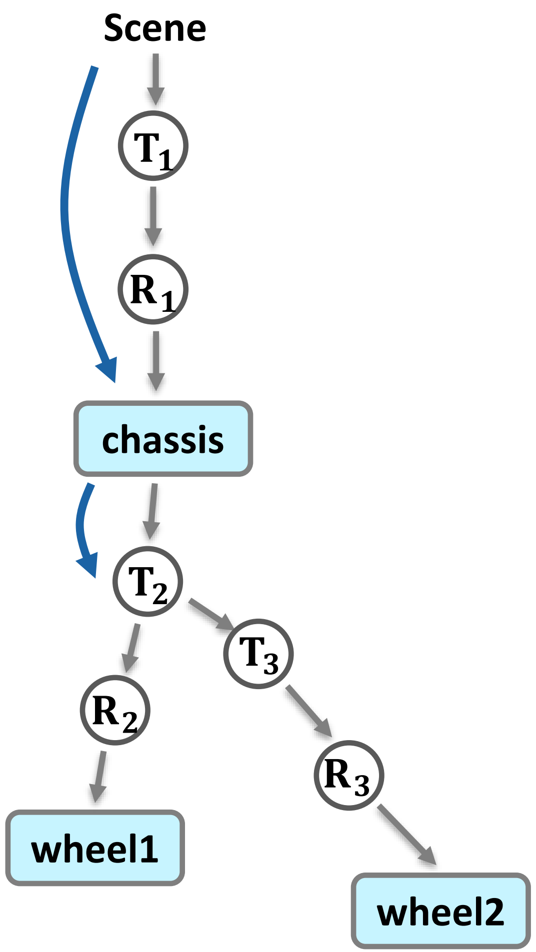


Need to visit all transformations
- Depth first traversal

Accumulate transformations when going down

Undo transformation on way up

Traversal of Transformations

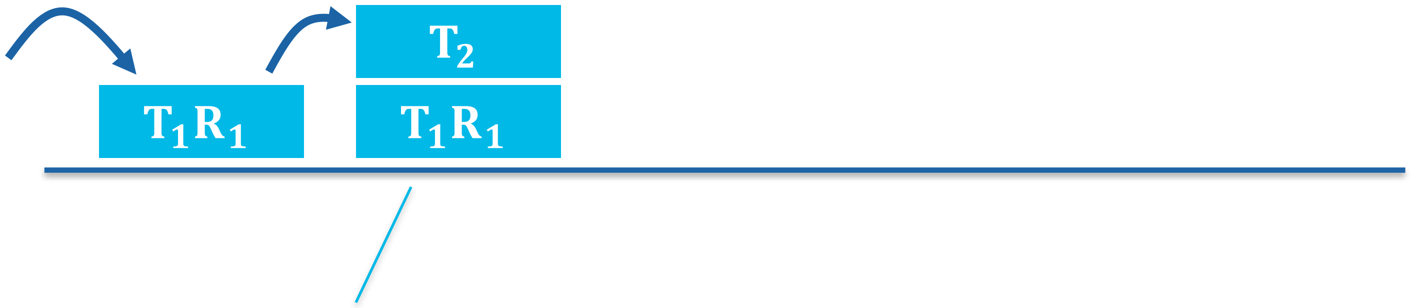


Need to visit all transformations

- Depth first traversal

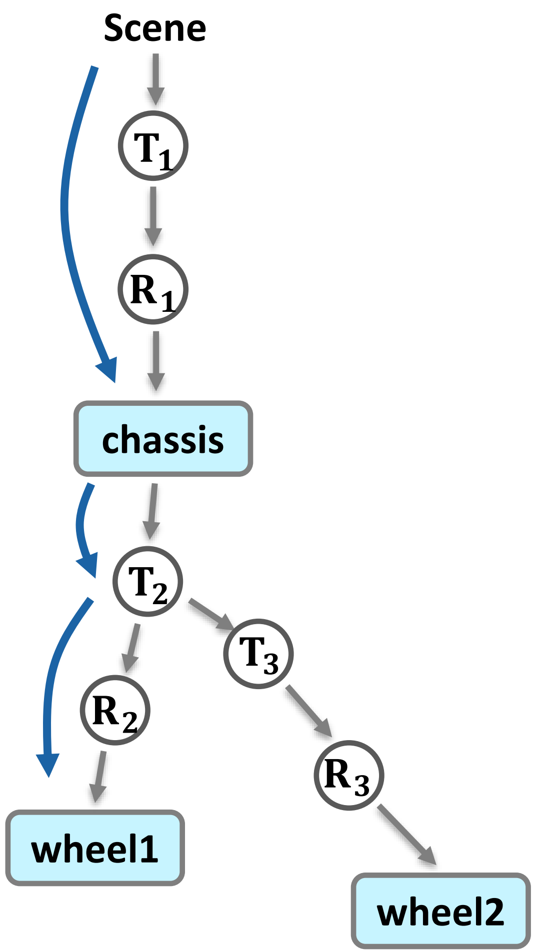
Accumulate transformations when going down

Undo transformation on way up



Stack of transformations
(read from top)

Traversal of Transformations

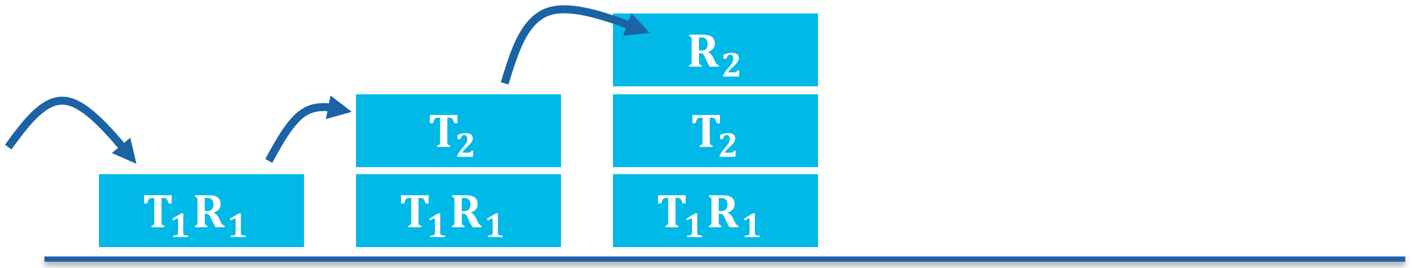


Need to visit all transformations

- Depth first traversal

Accumulate transformations when going down

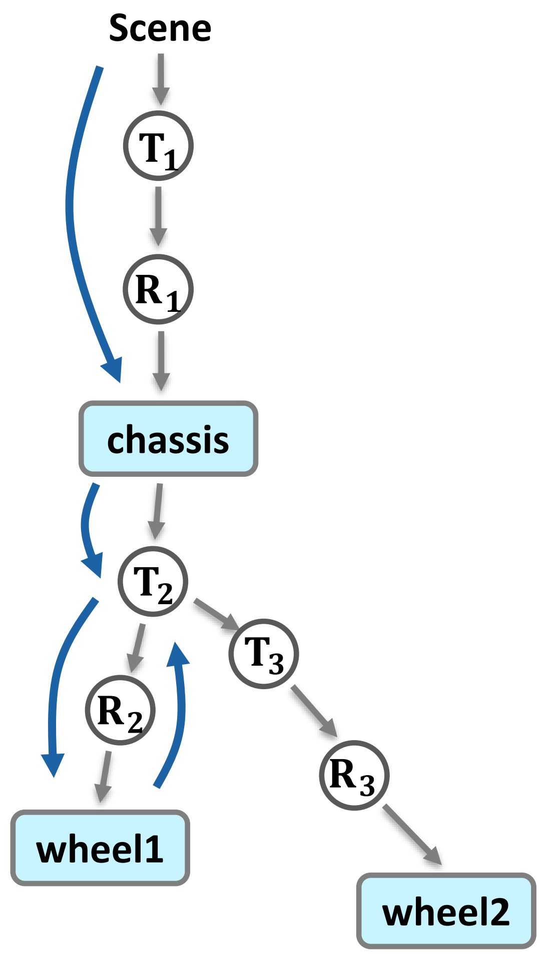
Undo transformation on way up



Stack of transformations (read from top)

$= \mathbf{T_1 R_1 T_2 R_2}$

Traversal of Transformations

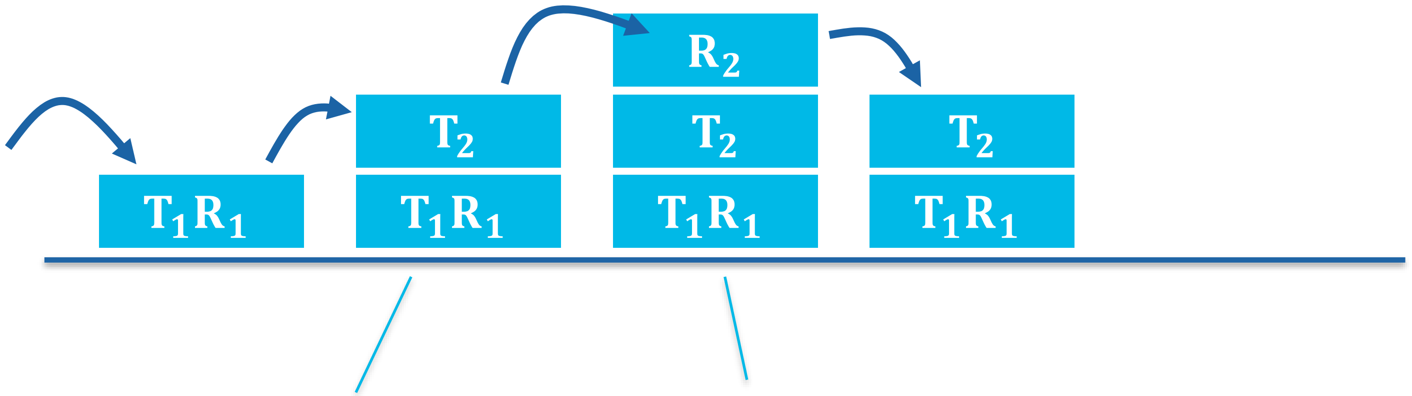


Need to visit all transformations

- Depth first traversal

Accumulate transformations when going down

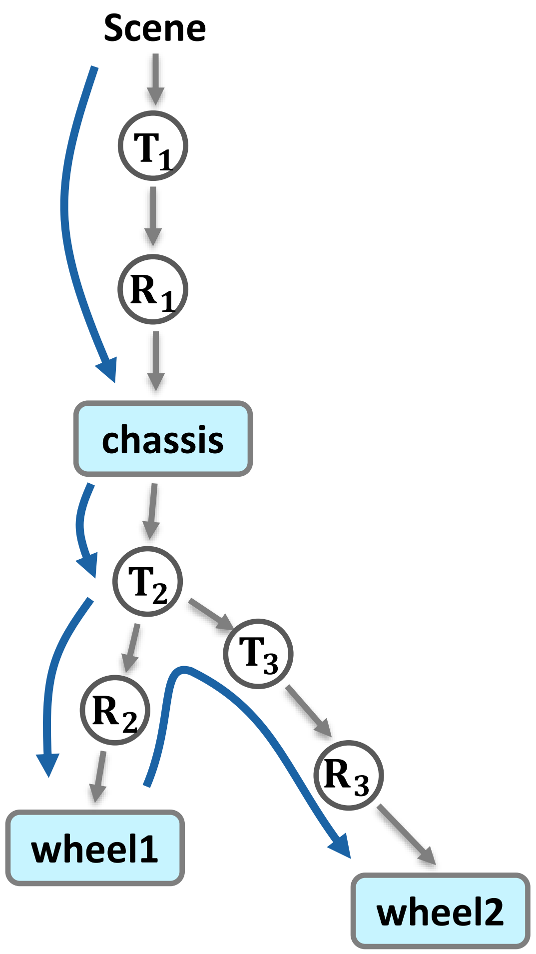
Undo transformation on way up



Stack of transformations (read from top)

$= T_1R_1T_2R_2$

Traversal of Transformations

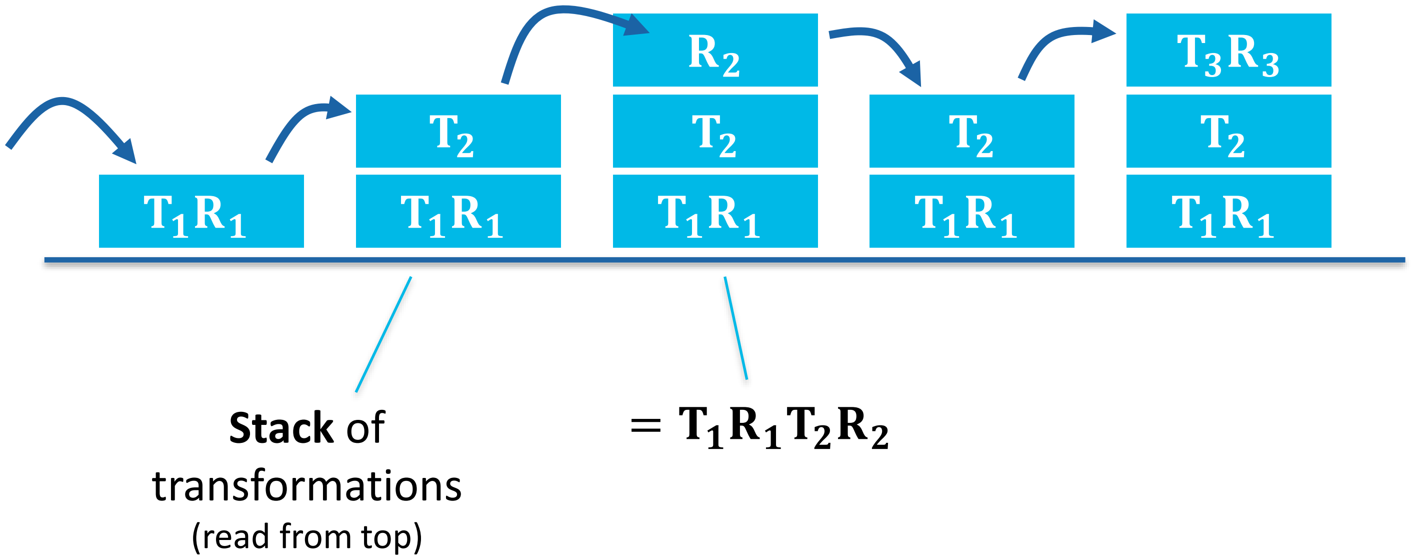


Need to visit all transformations

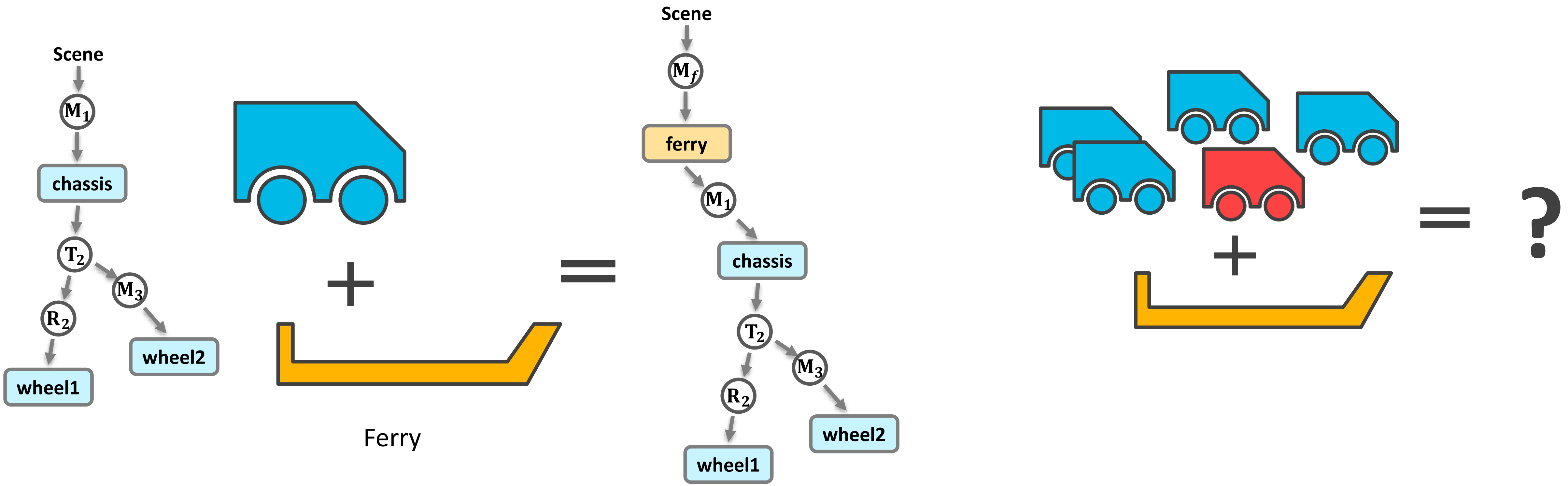
- Depth first traversal

Accumulate transformations when going down

Undo transformation on way up



Generalization



Adding more and more objects is quite cumbersome

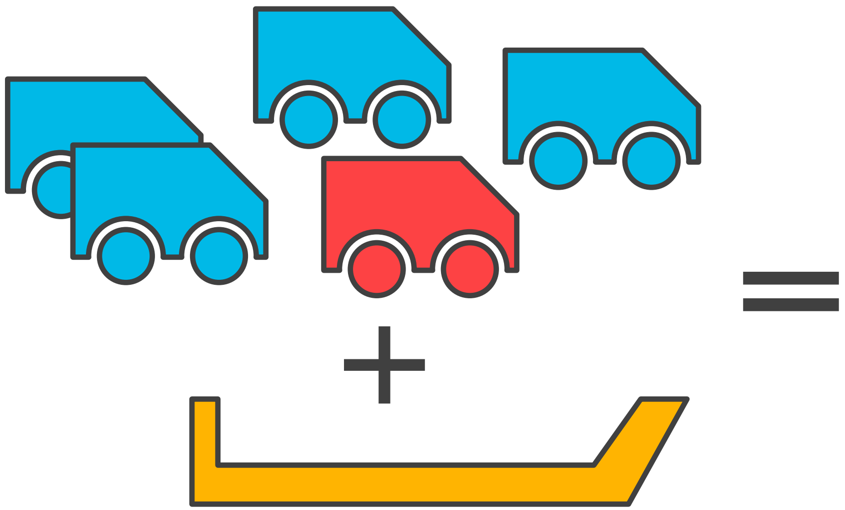
- Needs more than just hierarchical transformations

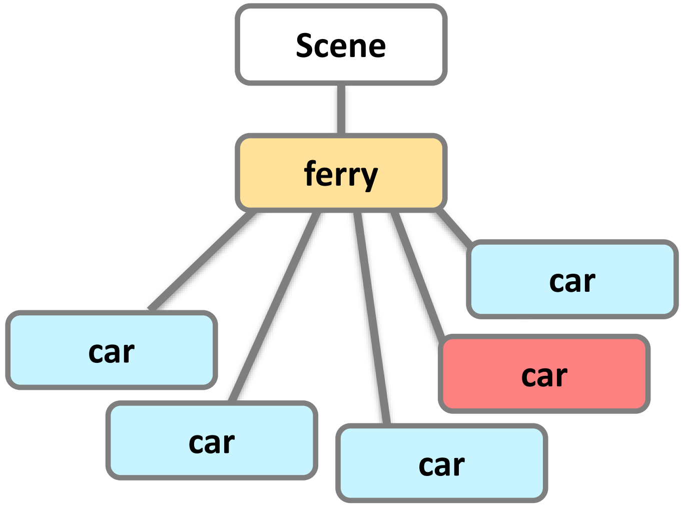
Generalization (cont.)

More general scene description?

What if the cars are identical?

- Is it possible to save some work?





Scene Graph

Graph structure representing a scene

- Nodes connected by edges

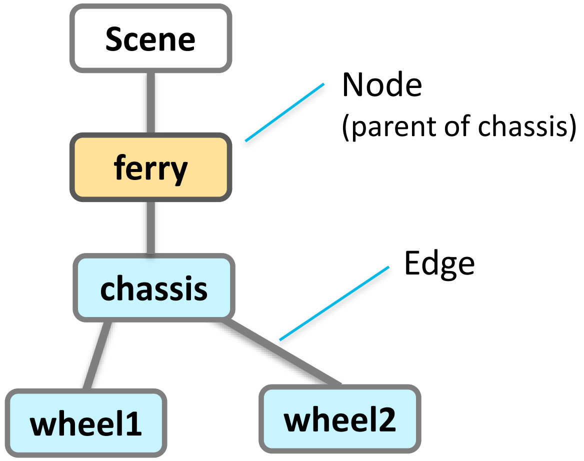
Simplest form: **tree**

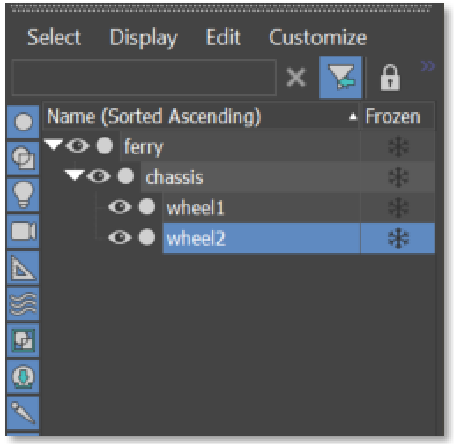
- One **root** node (scene)
- Each node has one parent

Transforms associated with edges **or** nodes

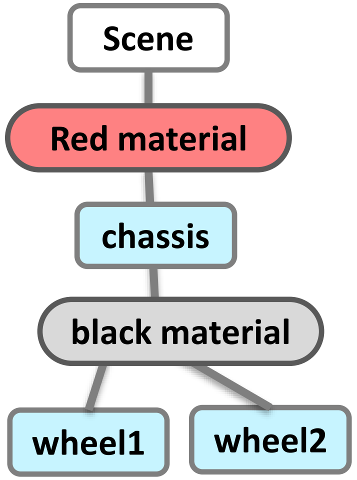
- Applies to geometry below
- Still hierarchical transformations!

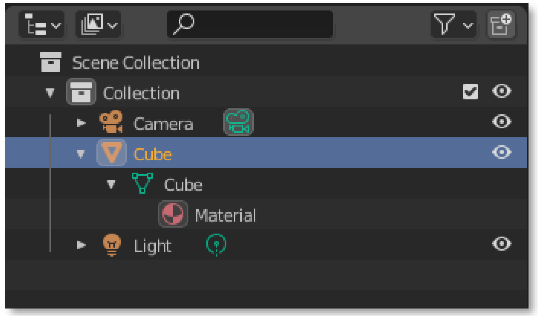
Materials, lights, cameras can all be part of graph





Scene graph in 3ds max





Scene graph in blender

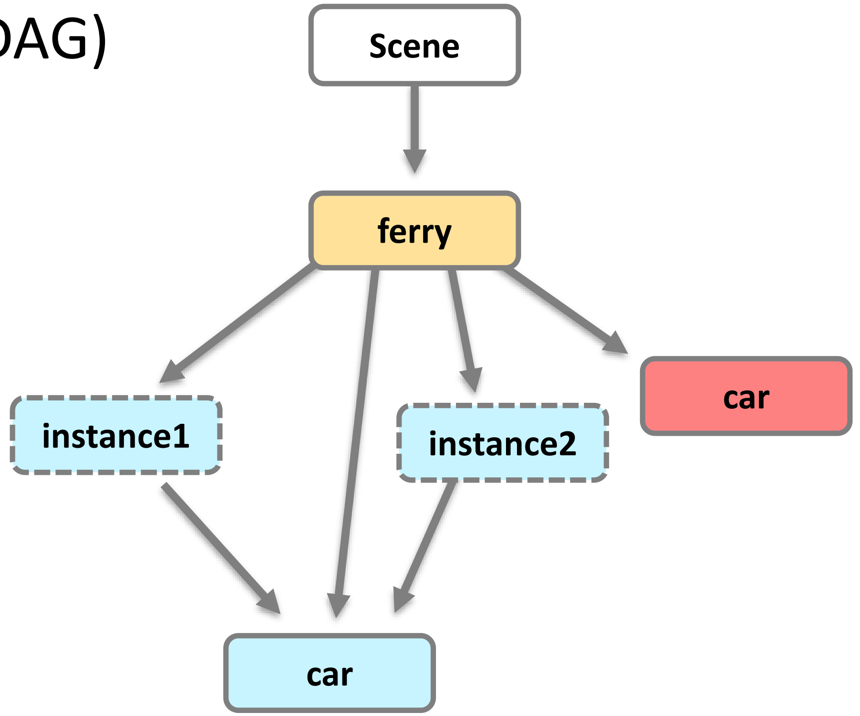
Scene Graph with Instances

Scene graph becomes a directed acyclic graph (DAG)

- No loops allowed!
- Groups cannot be part of themselves

Traversal more complicated

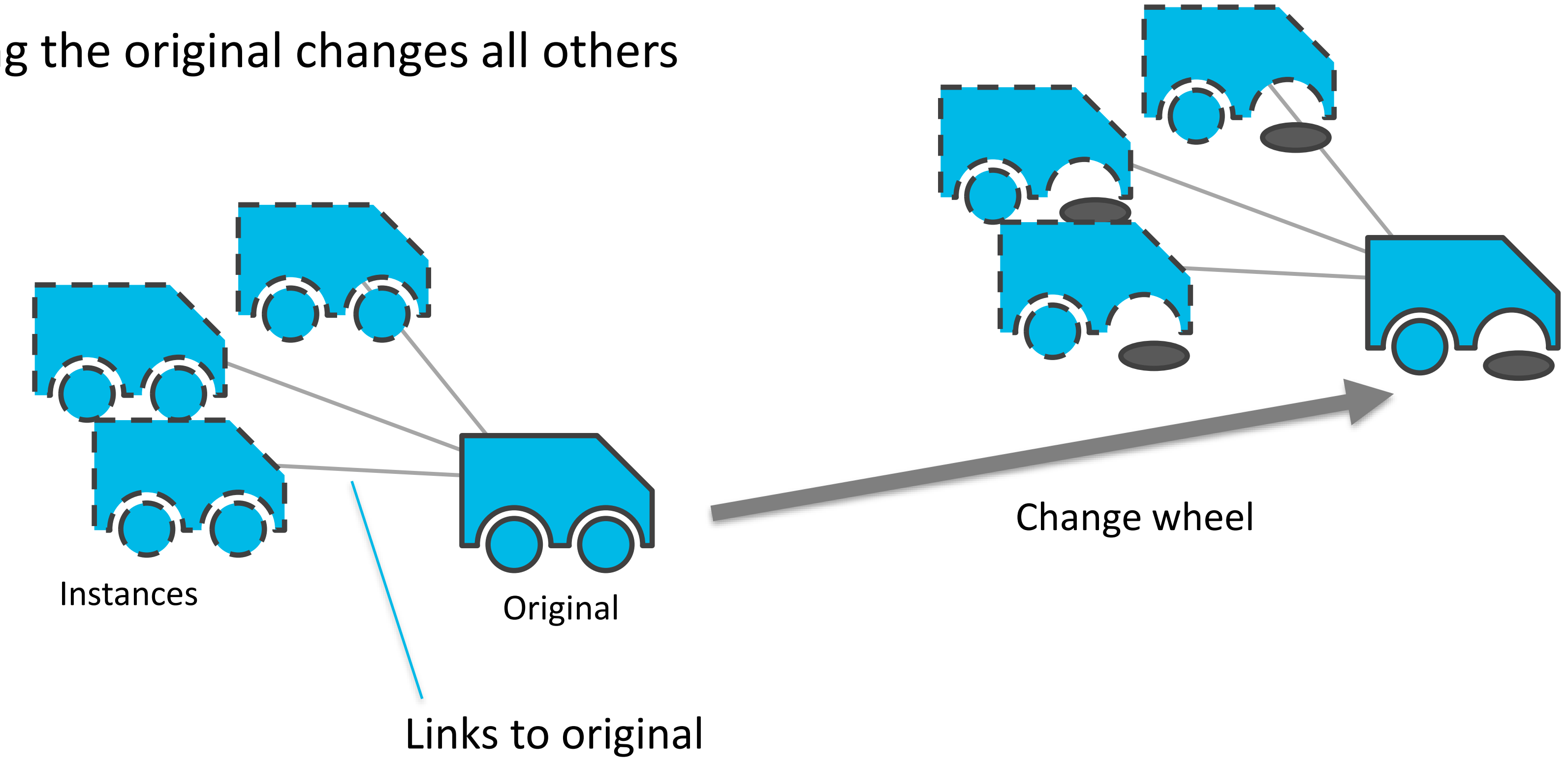
- Need to follow **all paths** from root node to leaves
- Transformations accumulated as before



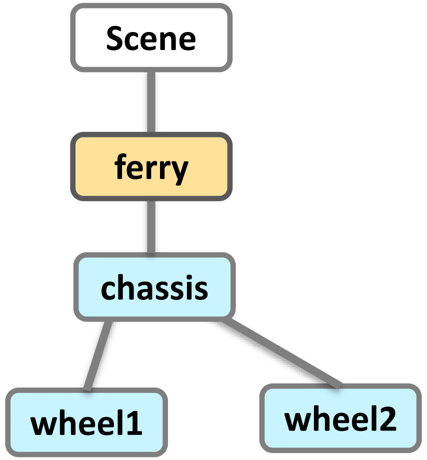
Object Instances

Instancing useful for identical objects

- Creates “clones” of an object
- Each one has own transformation
- Editing the original changes all others



Summary
– Scene Graphs –



Scene objects can be organized in a (hierarchical) scene graph

- A scene graph is a directed acyclic graph (DAG)
- Defines dependencies and parent-child relationships

Most basic form contains objects + transformations

- Possibility to store camera, lights, materials, …

Graph traversal possible using a transformation stack

Coming up next

Real time graphics and interaction

- Modern graphics hardware
- Rendering, shading
- Data structures for large scenes