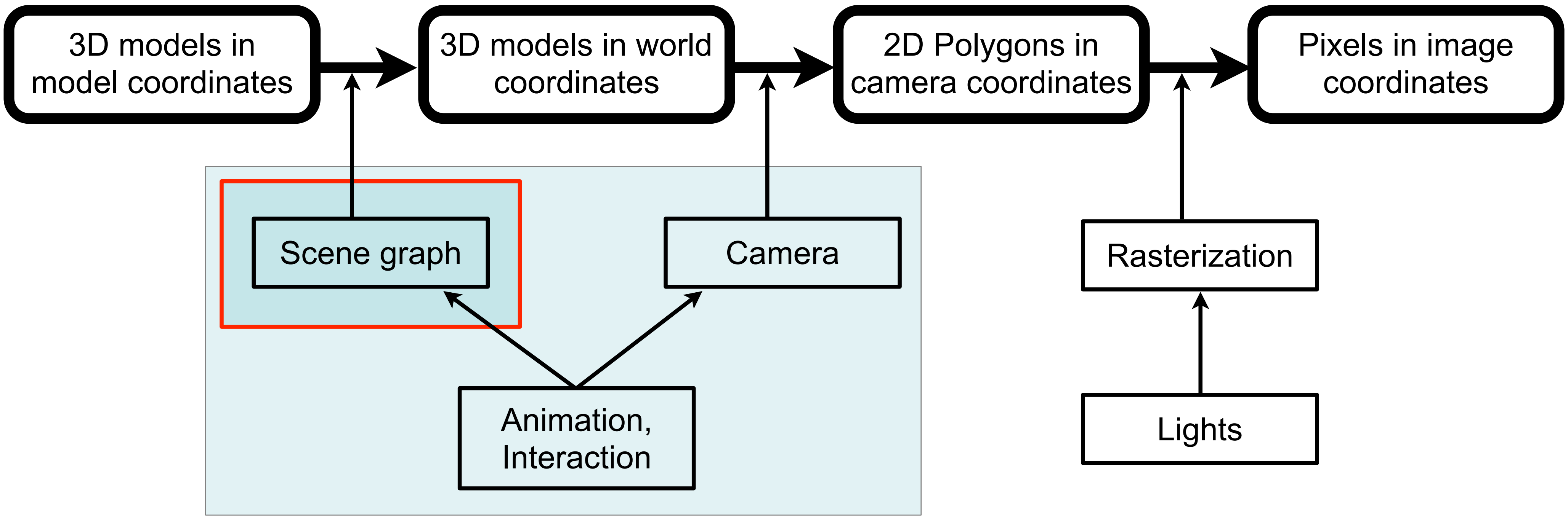
Chapter 6 - The Scene Graph

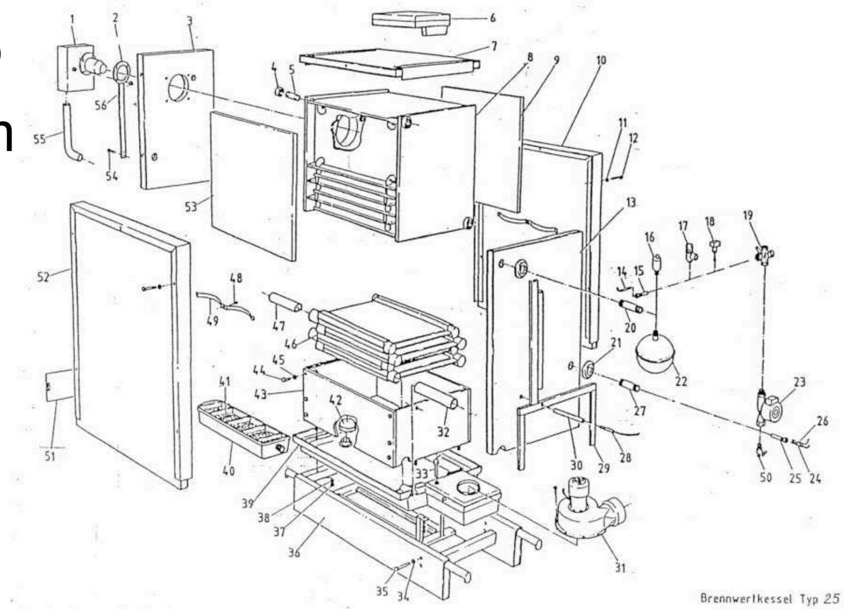
- Why a scene graph?
- What is stored in the scene graph?
  - objects
  - appearance
  - camera
  - lights
- Rendering with a scene graph
- Practical example

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, 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



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

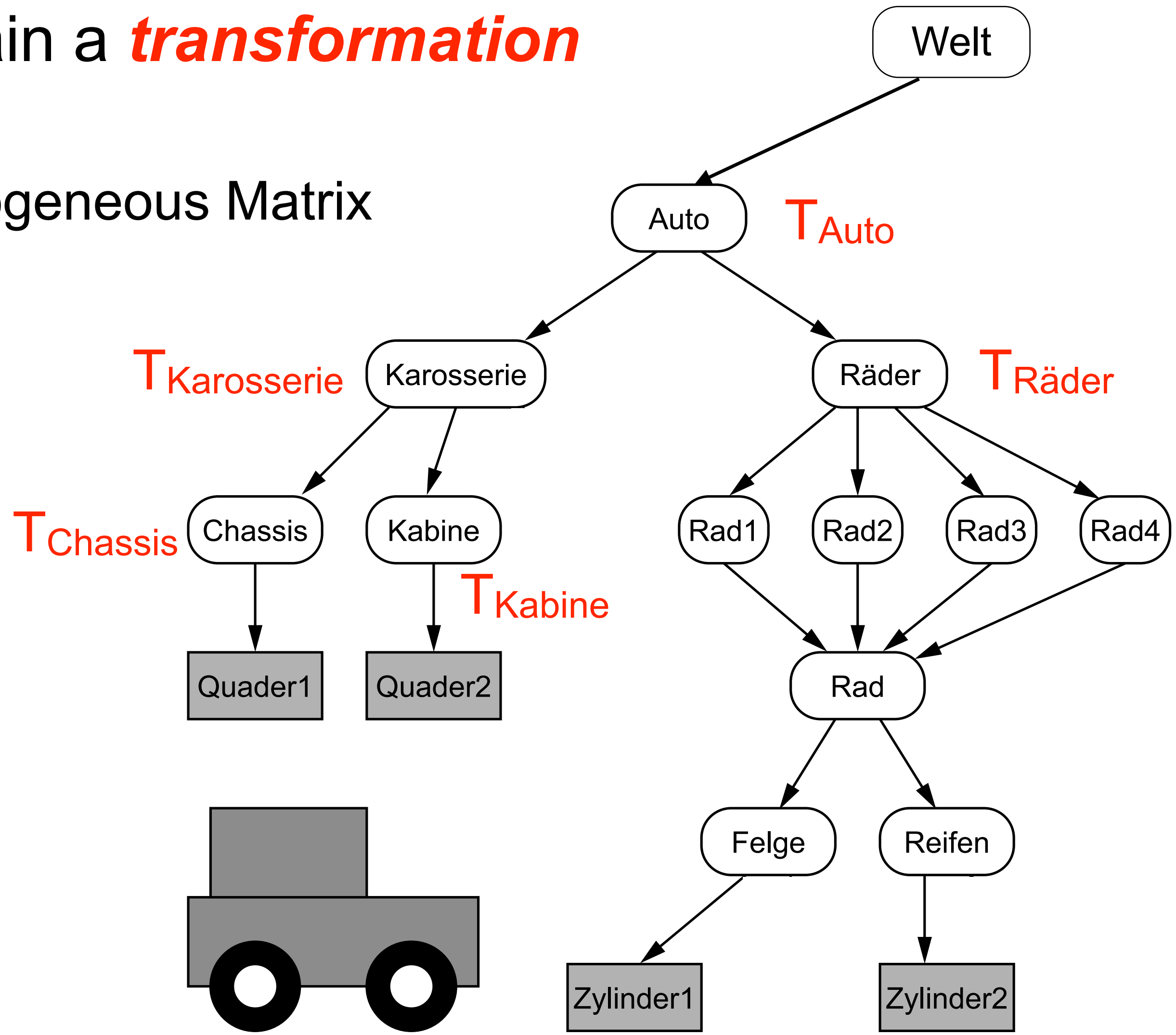
Chapter 6 - The Scene Graph

- 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

- Leafs 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

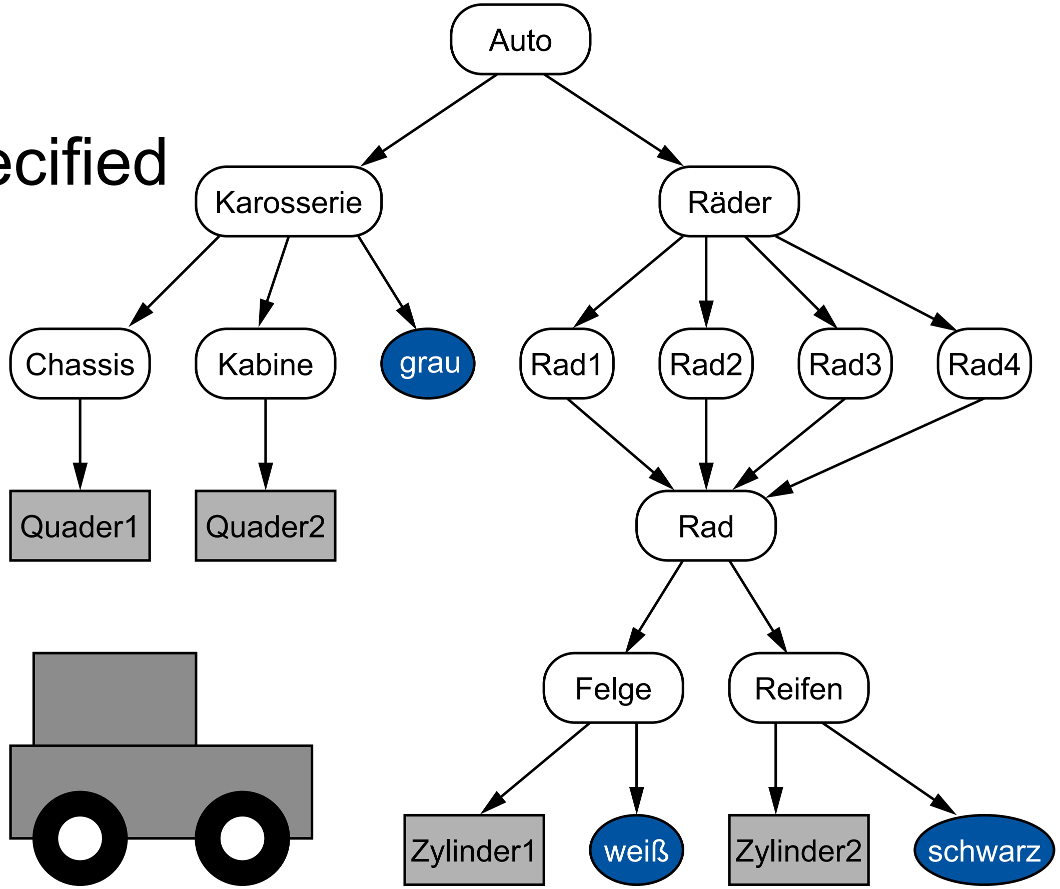
- Nodes can be the child of several groups
  - not a tree, but a directed acyclic graph (DAG)
  - effective reuse of geometry



Appearance in the Scene Graph

- Scene graph also contains appearances
  - Appearance: E.g. Color, reflection, transparency, texture
    Details see next lecture
  - can be reused similarly to geometry

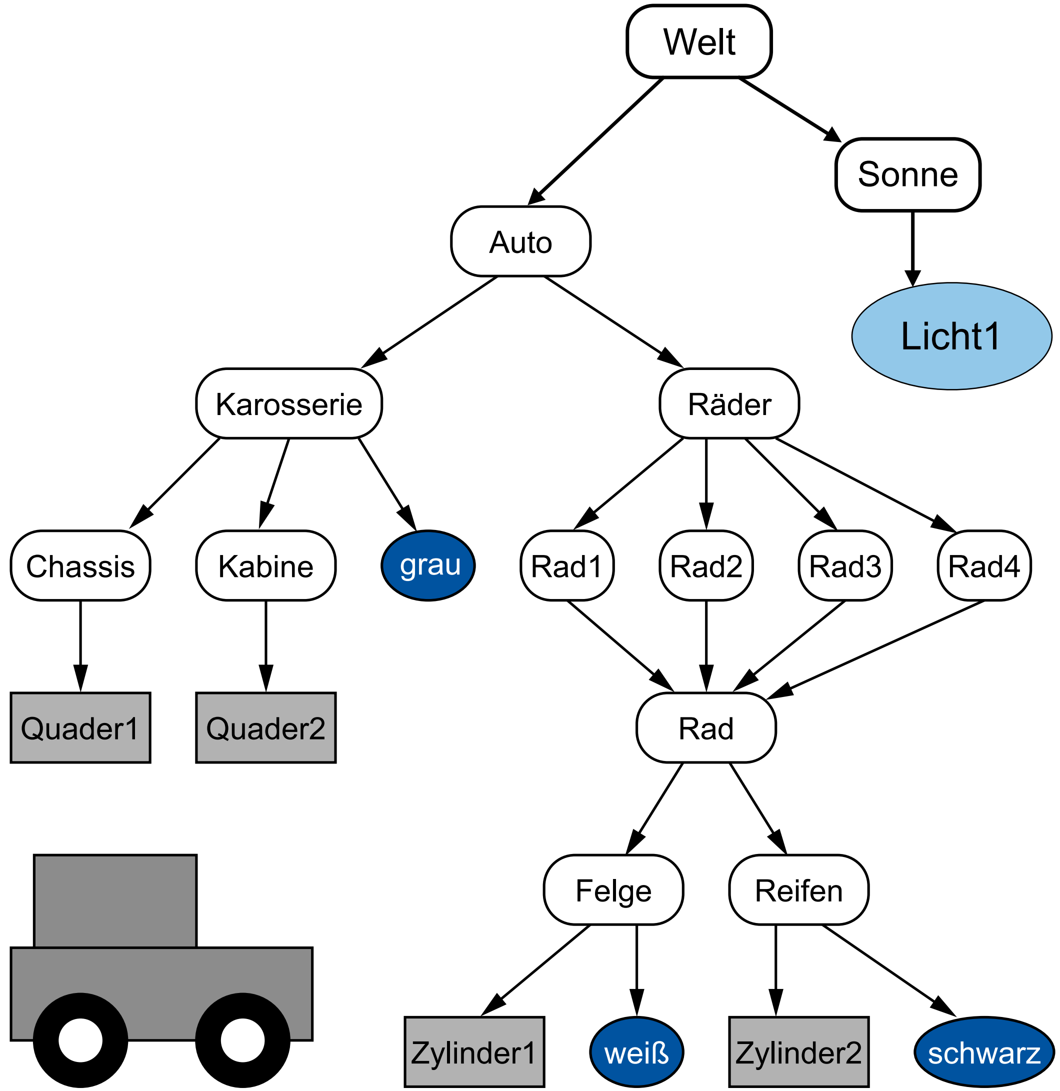
- Appearance can be only partially specified
  - unspecified values are inherited



Lights in the Scene Graph

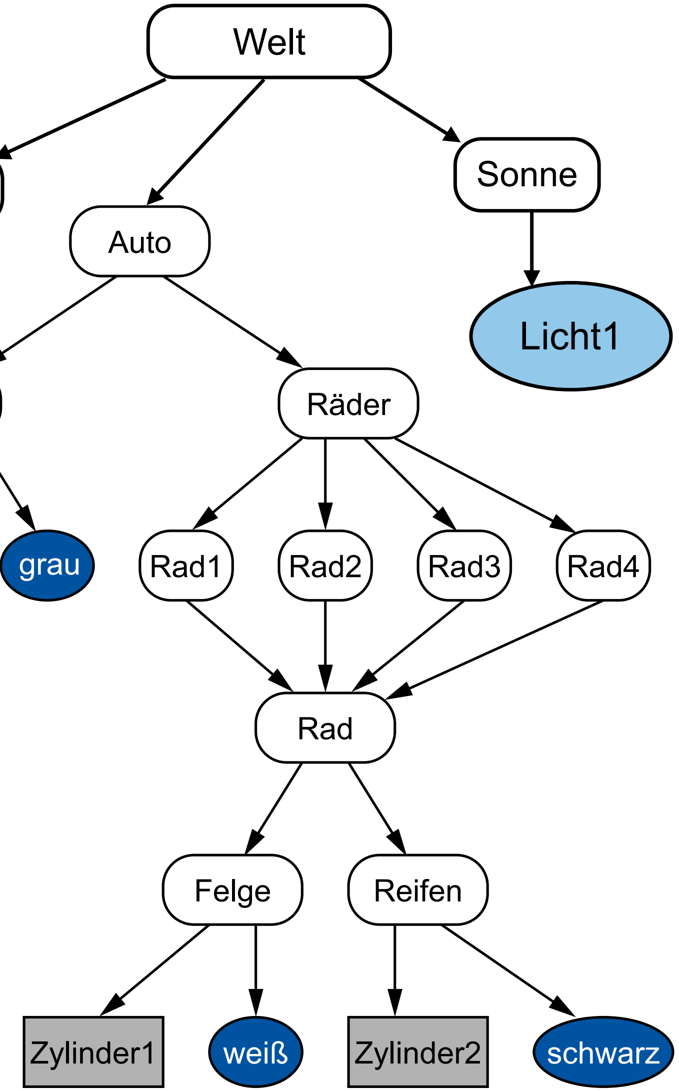
- Light sources also need a position and/or direction
  - Just include them into the scene graph
  - Can be animated just like geometry

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

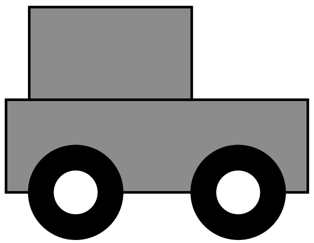


The Camera in the Scene Graph

- 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

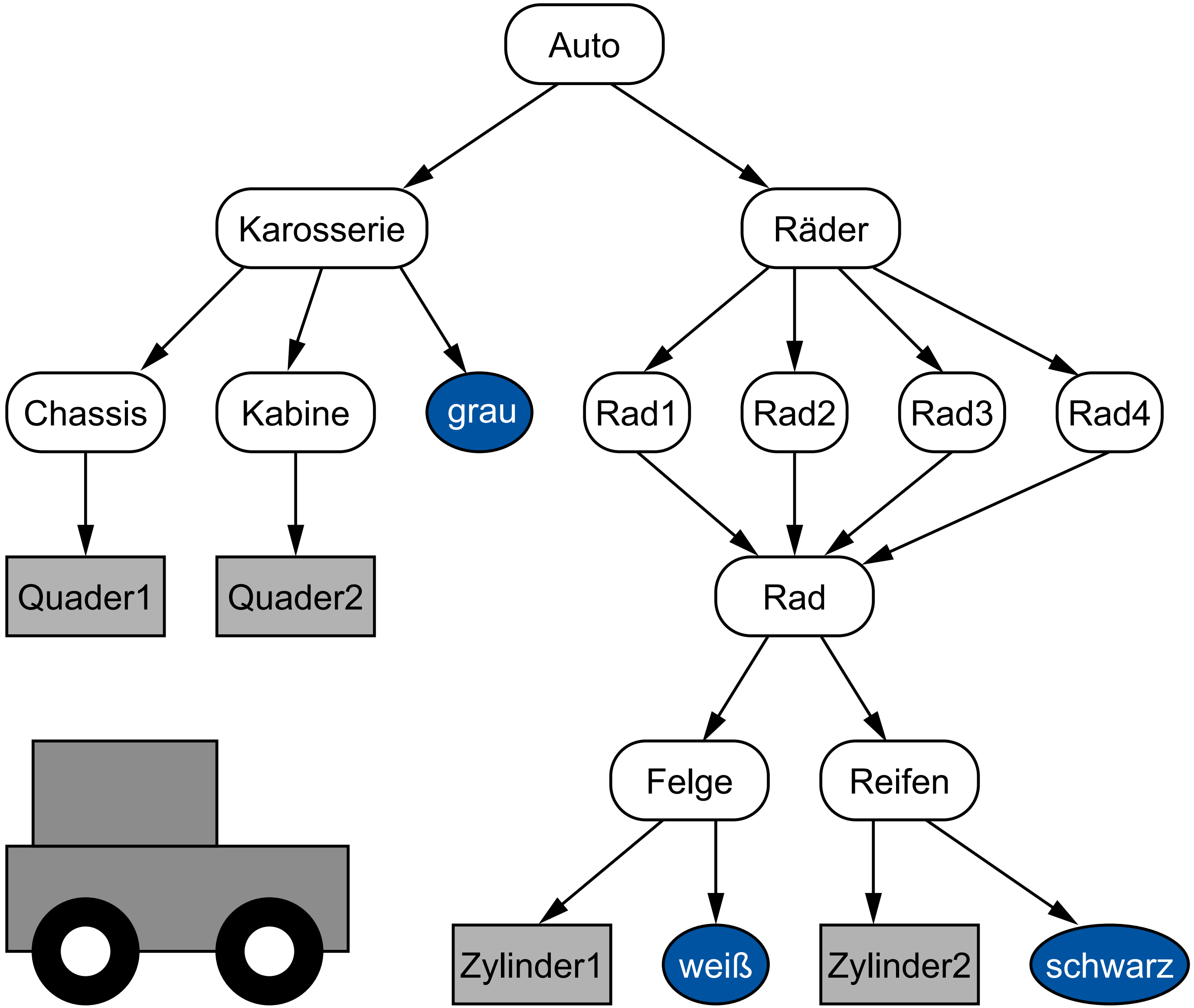


Chapter 6 - The Scene Graph

- Why a scene graph?
- What is stored in the scene graph?
  - objects
  - appearance
  - camera
  - lights
- Rendering with a scene graph
- Practical example

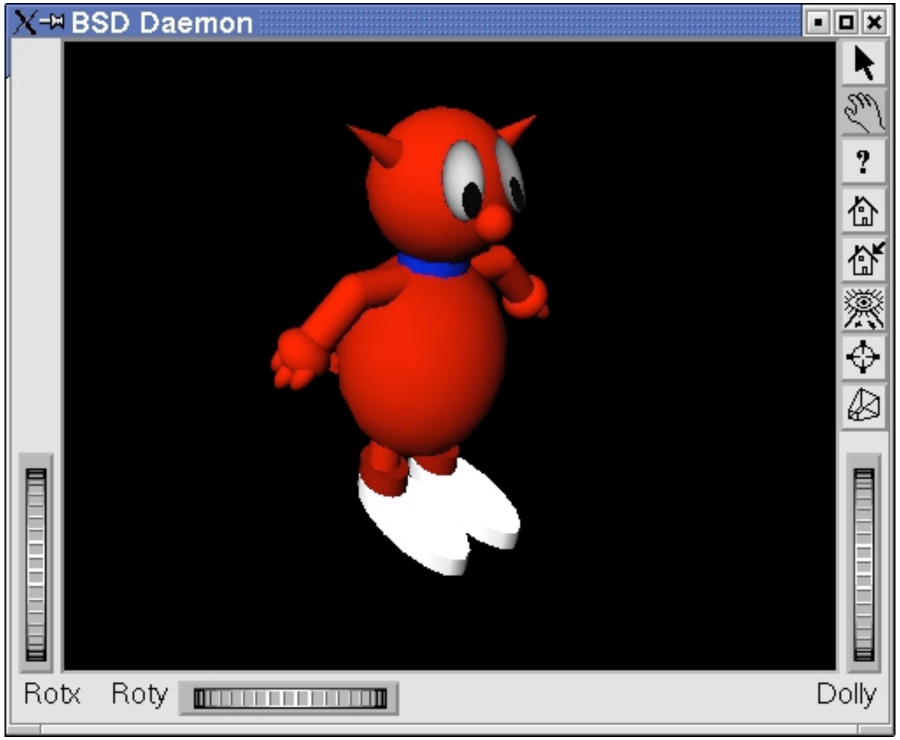
Scene graph traversal for rendering

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



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. JMonkey 3 (scene graph based game engine for Java)



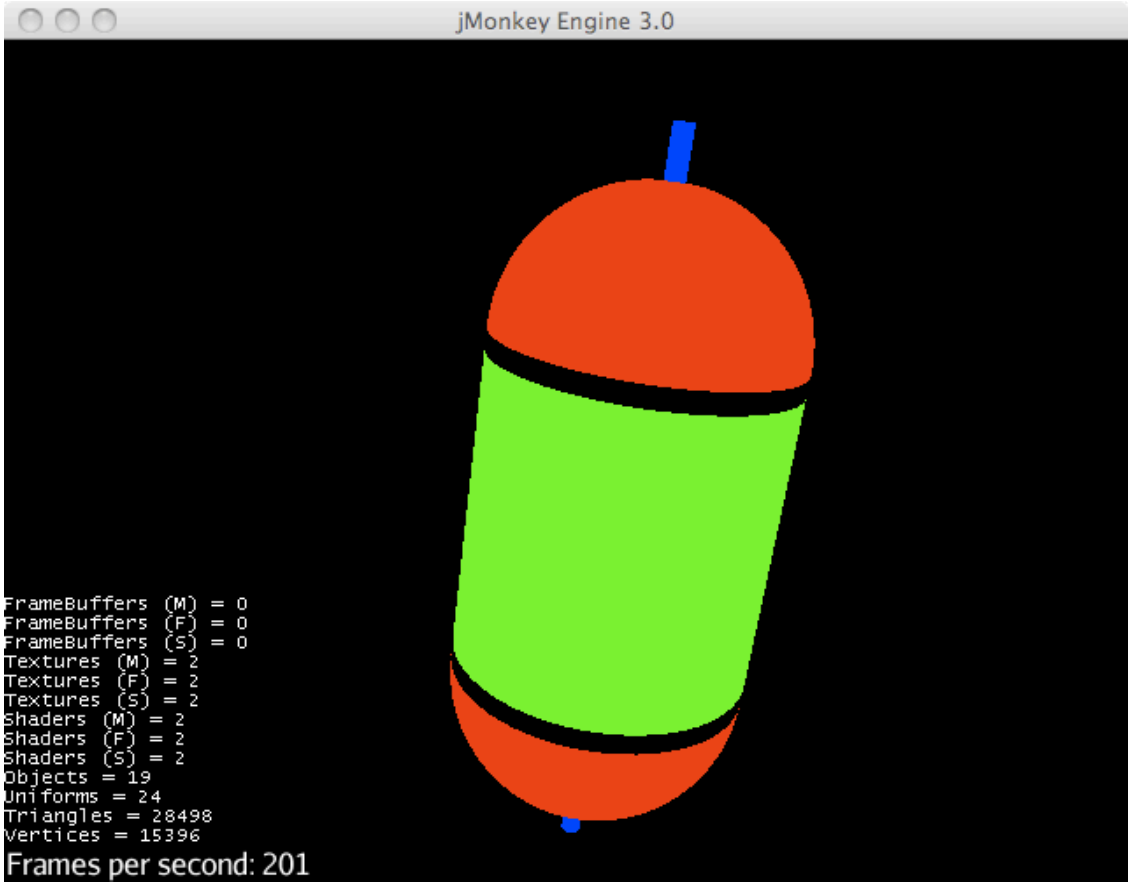
http://www.shlomifish.org/open-source/bits-and-bobs/open-inventor-bsd-daemon/

Chapter 6 - The Scene Graph

- Why a scene graph?
- What is stored in the scene graph?
  - objects
  - appearance
  - camera
  - lights
- Rendering with a scene graph
- Practical example

Example: Hierarchically Structured Object

- Simple object composed from basic geometric forms
  - Here: Cylinders, hemispheres
  - Resembling e.g. a resistor in electronics
- Main form element: Cylinder
- Additional form elements: Two ends
  - Each end consisting of:
    - a hemisphere
    - a connector



Exercise: Scene Graph for Example

- What is the scene graph for the example?
- Which are the transformations in each node?
- Which additional information should be stored in each node?

Example Implementation

- Using “JMonkey Engine 3” (JME3)
  - Open Source community project
  - Based on Java and OpenGL
  - Using scene graphs as core concept (as most gaming engines)
  - See http://jmonkeyengine.org/
- Terminology of the JMonkey scene graph:
  - “Spatial”: Common abstraction for all nodes in a scene graph
  - “Node”: Abstract (inner) nodes in a scene graph – not rendered
  - “Geometry”: Leaf node in a scene graph – visibly rendered
- Information attachable to scene graph nodes (“spatials”):
  - Local affine transformation (translation, scaling, rotation)
  - Material (e.g. self-illuminating colored materials, but many else)
    - see next lecture

Local Coordinates and World Coordinates

- Each primitive object is created in a local coordinate system
  - Around the origin or at a specified location
- Object is moved/scaled/rotated to required position
  relative to father node next level up
- Object is inserted into scene graph
  - actually determines father node
- World coordinate position of object
  - is determined by composition of all transformations along path from root to object
  - as used in rendering algorithm
- Objects:
  - simple geometrical objects in this section
  - general polygon meshes (see last chapter) in practice

Example (JMonkey Engine) Part 1

- Creating the core part of the scene
  - Cylinder constructor in JME: (*samples in axis, samples in radius, radius, height*)
    - What does this mean?
    - "Mesh": pure geometrical data, to be wrapped into scene graph objects
  - Why is it likely that we need a rotation for seeing the object like we want it?
    - Around which axis? What is the unit for the angle?

```
/** create a green cylinder at origin */
Cylinder cylMesh = new Cylinder(64,64,1.5f,3);
Geometry cylinder = new Geometry("Cylinder", cylMesh);
Material mat1 = new Material(assetManager,
        "Common/MatDefs/Misc/Unshaded.j3md");
mat1.setColor("Color", ColorRGBA.Green);
cylinder.setMaterial(mat1);
cylinder.rotate(90*FastMath.DEG_TO_RAD,0f,0f);
```

Example (JMonkey Engine) Part 2

```
/** create a red dome  */
Dome domeMesh = new Dome(Vector3f.ZERO,64,64,1.5f,false);
Geometry dome1 = new Geometry("UpperDome", domeMesh);
Material mat2 = new Material(assetManager,
        "Common/MatDefs/Misc/Unshaded.j3md");
mat2.setColor("Color", ColorRGBA.Red);
dome1.setMaterial(mat2);
```

- This creates a new object:
  - Where is it located by default?
  - What do we have to do with it to make good use of it?

Example (JMonkey Engine) Part 3

```
//++ create a little blue cylinder */
Cylinder litCylMesh = new Cylinder(32,32,0.1f,1);
Geometry litCylinder = new Geometry("Cylinder", litCylMesh);
Material mat3 = new Material(assetManager,
        "Common/MatDefs/Misc/Unshaded.j3md");
mat3.setColor("Color", ColorRGBA.Blue);
litCylinder.setMaterial(mat3);
litCylinder.rotate(90*FastMath.DEG_TO_RAD,0f,0f);
litCylinder.move(0f,1.5f,0f);

/** upper end: combine red dome and little blue cylinder by node */
Node upperEnd = new Node("upperEnd");
upperEnd.attachChild(dome1);
upperEnd.attachChild(litCylinder);
```

- What is the overall result (“Upper End”) of this?
- Why is it important to move the little cylinder before combining it with the red dome?

Example (JMonkey Engine) Part 3

```
/** lower end: create a clone of upper end */
Node lowerEnd = (Node) upperEnd.clone();

/** put the upper end above the cylinder */
upperEnd.move(0f,1.7f,0f);

/** put the lower end below the cylinder */
lowerEnd.move(0f,-1.7f,0f);
lowerEnd.rotate(180*FastMath.DEG_TO_RAD,0f,0f);
```

- Why is this program code so short?
  - Why is it good to use a “clone” function here?

Example (JMonkey Engine) Part 4

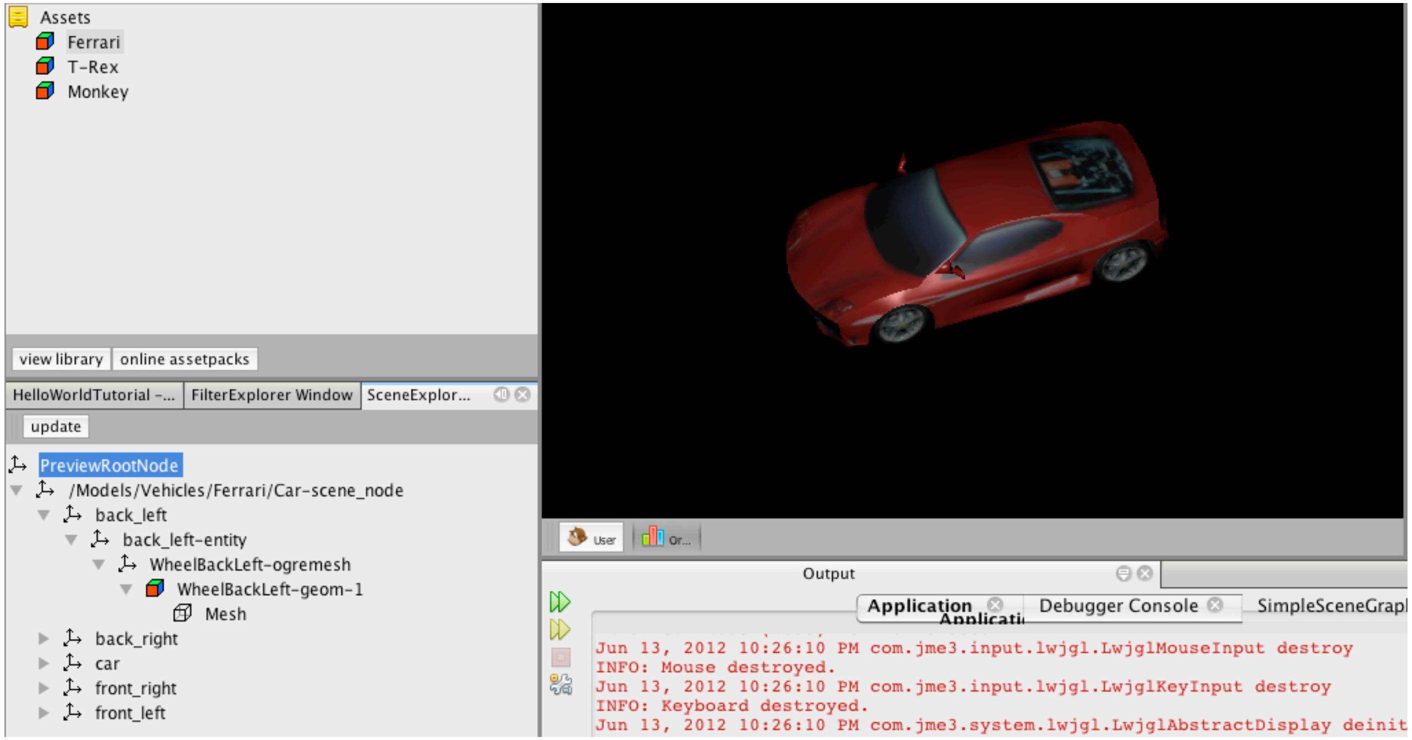
- Overall presentation
  - Compose main objects of the scene
  - Attach scene objects to world root (“rootNode” in JME)
  - Carry out global transformations for whole world
- Projection modes (e.g. orthographic vs. perspective)
  - may be specified at this level
- Camera position (and other camera parameters)
  - may be specified separately for projection or may be part of scene graph

```
 /** Create a pivot node at (0,0,0) and attach it to the root node */
 Node pivot = new Node("pivot");
 rootNode.attachChild(pivot); // put this node in the scene

 pivot.attachChild(cylinder);
 pivot.attachChild(upperEnd);
 pivot.attachChild(lowerEnd);
/** Rotate the pivot node: Note that all objects have rotated! */
 pivot.rotate(0.4f,0.4f,0f);
```

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:
  - JME Scene



Outlook: Lighting and Scene Graphs

- Types of light:
  - Ambient light: No specific direction, like diffuse day light
  - Directional light:
    - No specific source location, but specific direction
    - Like sunlight
  - Various artificial light sources (spot lights, point lights):
    - Occupy specific position in scene graph
- Effect of light depending on material
- See next lecture