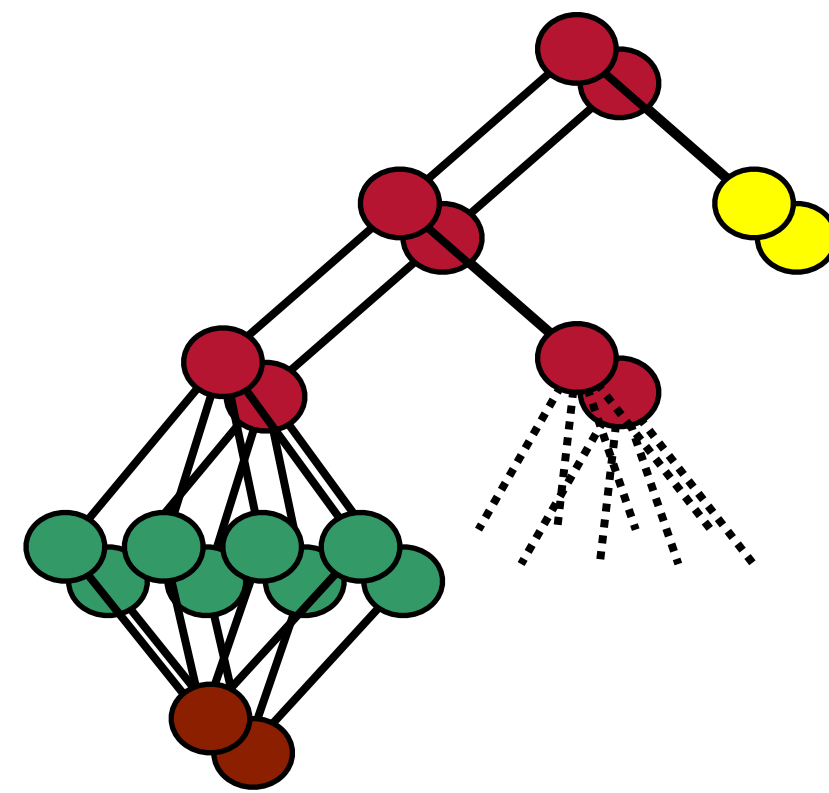
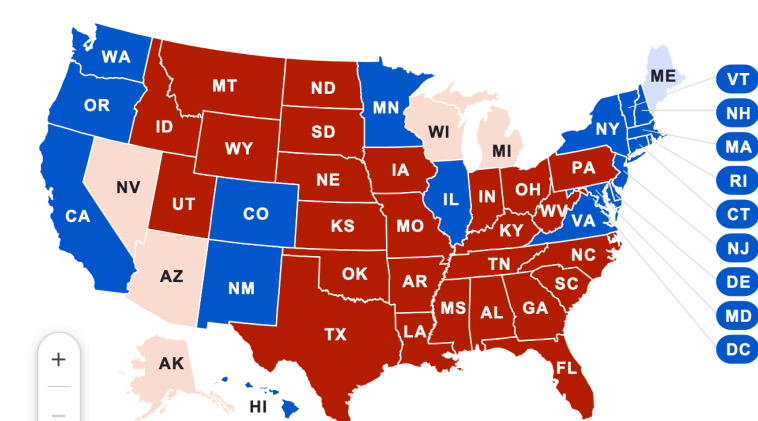
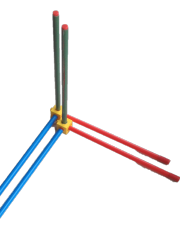


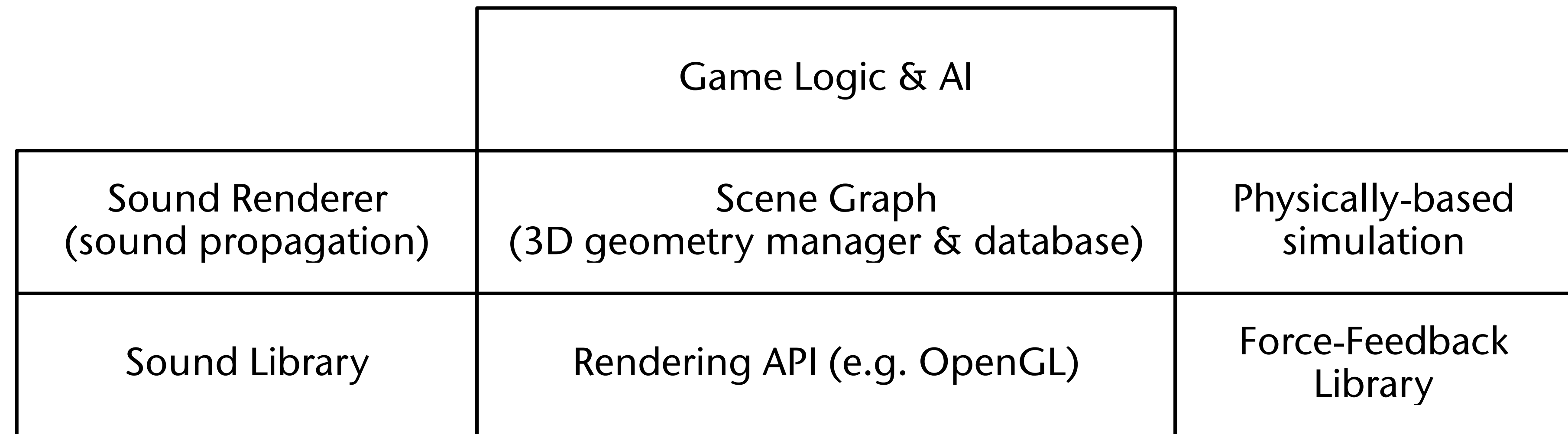
Virtual Reality & Physically-Based Simulation Scenegraphs & Game Engines



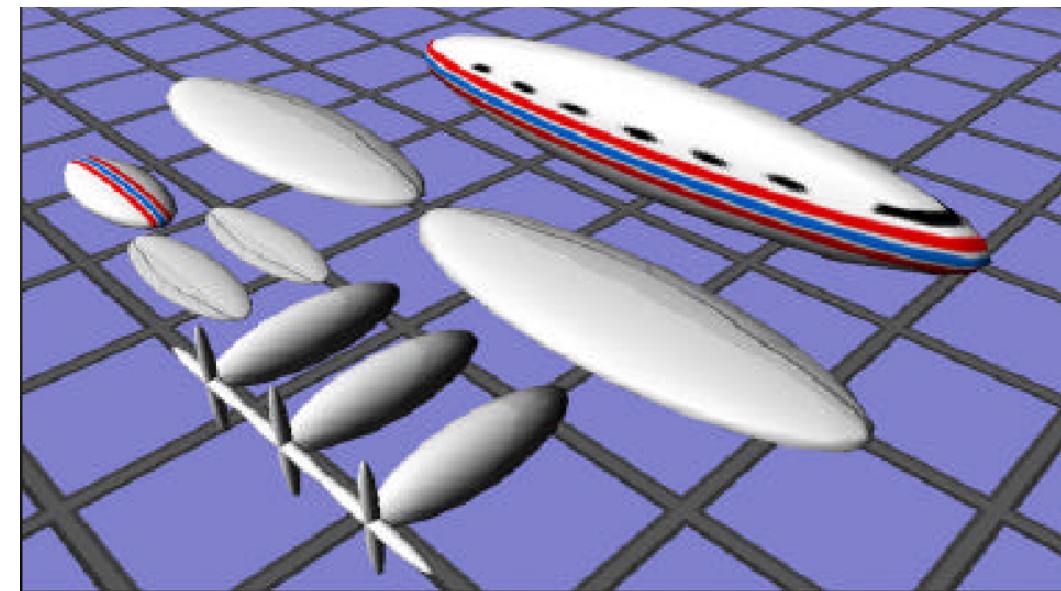
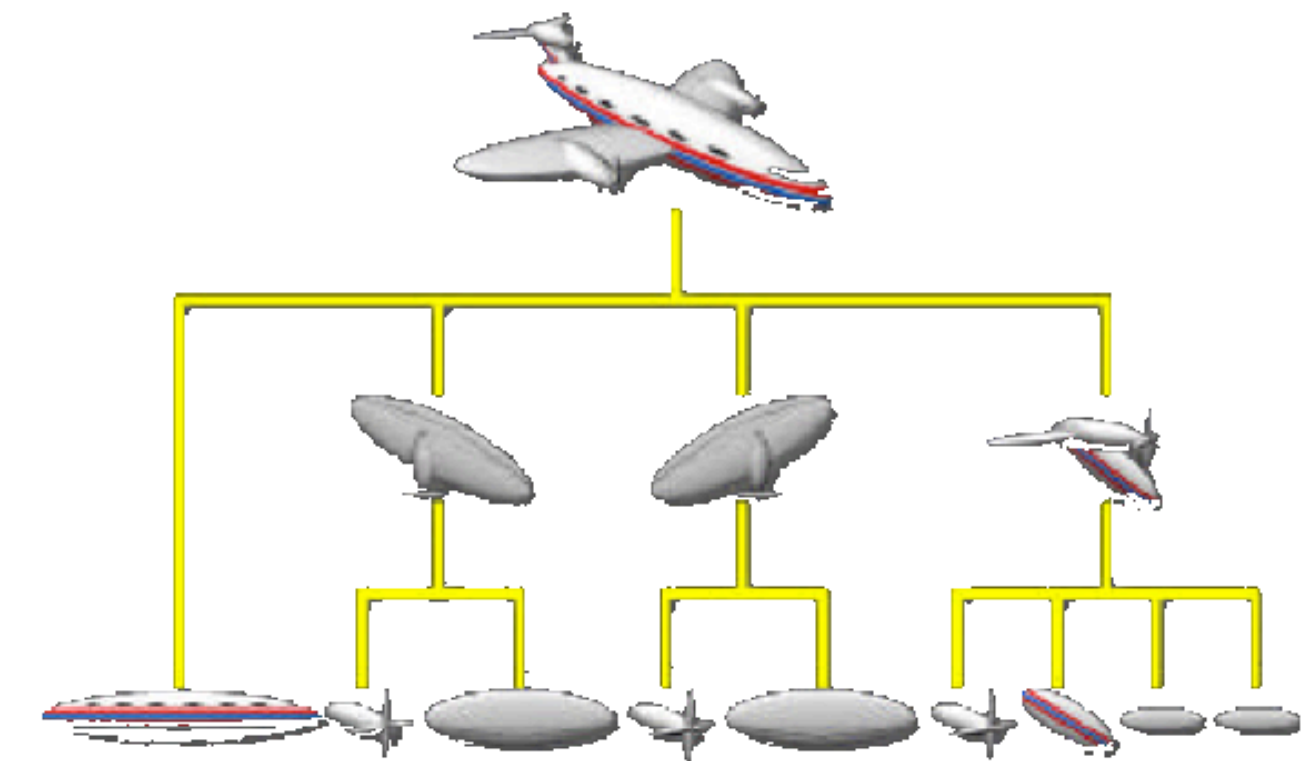
G. Zachmann
University of Bremen, Germany
cgvr.cs.uni-bremen.de



Overall VR System Architecture (Example)

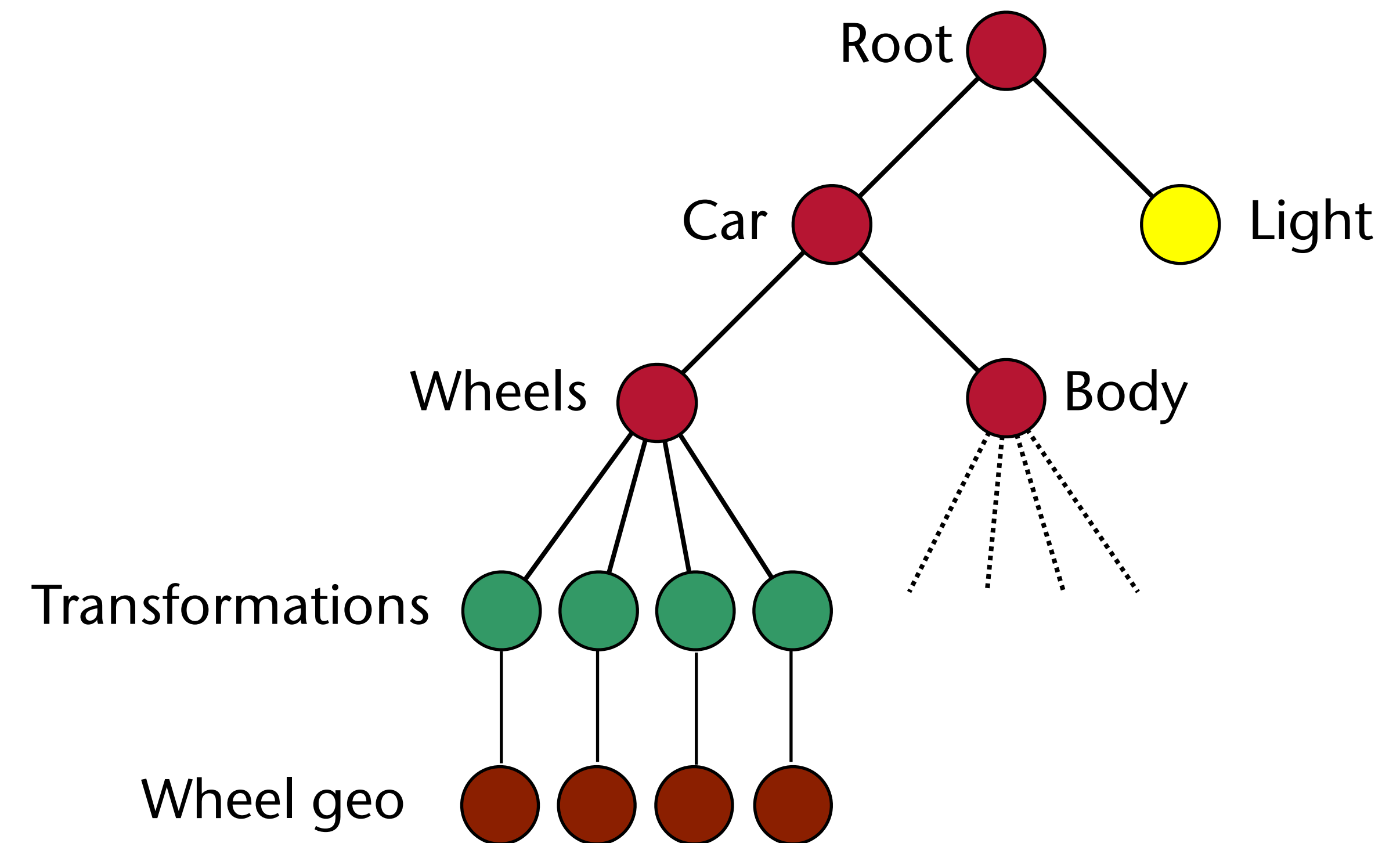


- Immediate mode vs. retained mode:
 - Immediate mode = OpenGL / Direct3D = Application sends polygons / state change commands to the GPU → more flexible
 - Retained mode = scenegraph = applications builds pre-defined data structures that store polygons and state changes → easier and (probably) more efficient rendering
- Flat vs. Hierarchical scene descriptions:
- *Code re-use and knowledge re-use!*
- Descriptive vs. imperative (cv. Prolog vs. C)
 - Thinking objects ... not rendering processes



Structure of a Scene Graph

- Directed, acyclic graph (DAG)
 - Often even a proper tree
- Consists of heterogeneous nodes
- Example:



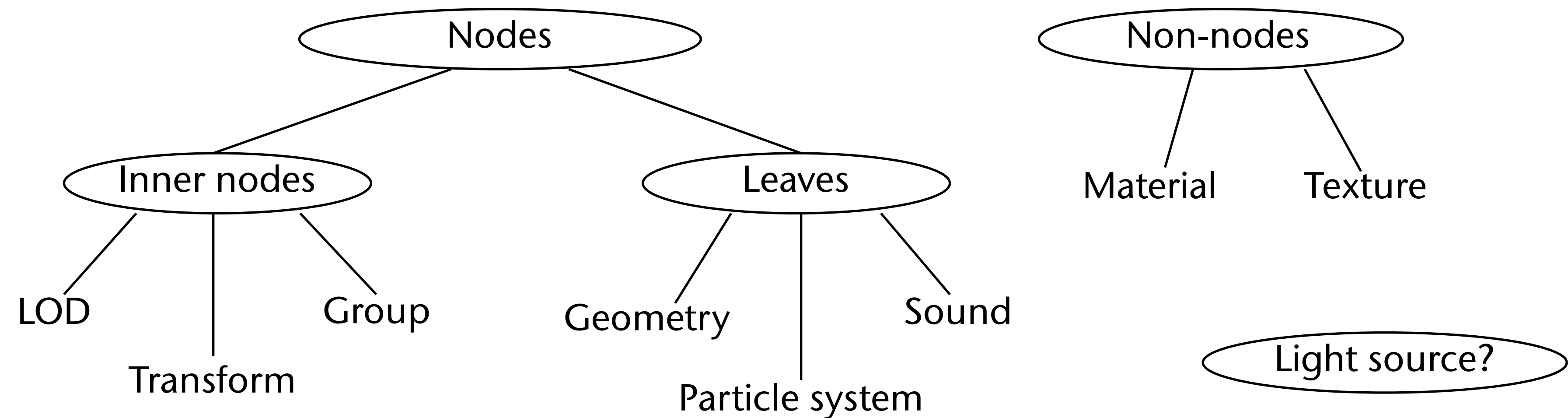
- Most frequent operation on scene graph: rendering
 - Amounts to depth-first traversal
 - Operation per node depends on kind of node

Semantics of the Elements of a Scenegraph

- Semantics of a node:
 - Root = "universe"
 - Leaves = "content" (geometry, sound, ...)
 - Inner nodes = forming groups, store state (changes), and other non-geom. functionality, e.g., transforms
- Grouping: criteria for grouping is left to the application, e.g., by
 - Geometric proximity → scenegraph induces a nice BVH
 - Material → good, because state changes cost performance!
 - Meaning of nodes, e.g., all electrical objs in the car under one node → good for quickly switching off/on all electrical parts in the car
- Semantics of edges = inheritance of states
 - Transformation
 - Material
 - Light sources (?)

Kinds of Nodes

- There are 2 hierarchies: scenegraph hierarchy + class hierarchy
- The flexibility and the expressiveness of a scenegraph depends heavily on the kinds and number of node classes!



Special Elements of a Scene

- Light sources:
 - Usually part of the scenegraph
 - Problem with naïve semantics: what if light source should move/turn, but not the scene it shines on?
 - Solution: **beacons**
 - Light source node lights its sub-scene underneath
 - Position/orientation is taken from the beacon

- Camera: to be, or not to be a node in the scenegraph?
 - Both ways have dis-/advantages
 - If not a node: use beacon principle

