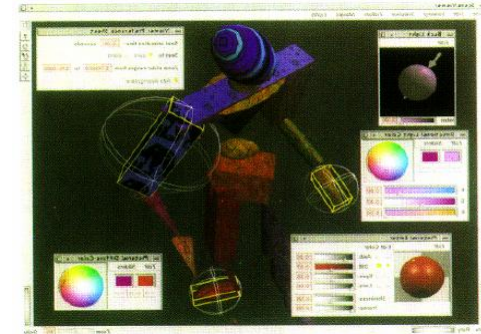
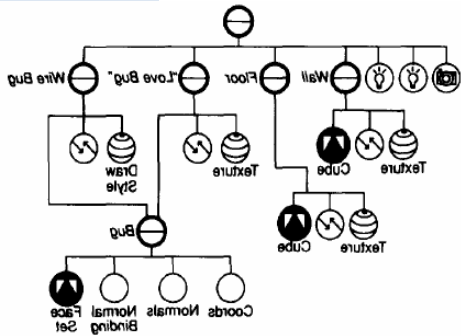


High Level Graphics Programming

Hannes Kaufmann



Interactive Media Systems Group (IMS)
 Institute of Software Technology and
 Interactive Systems

VR & AR Course Overview

- Introduction

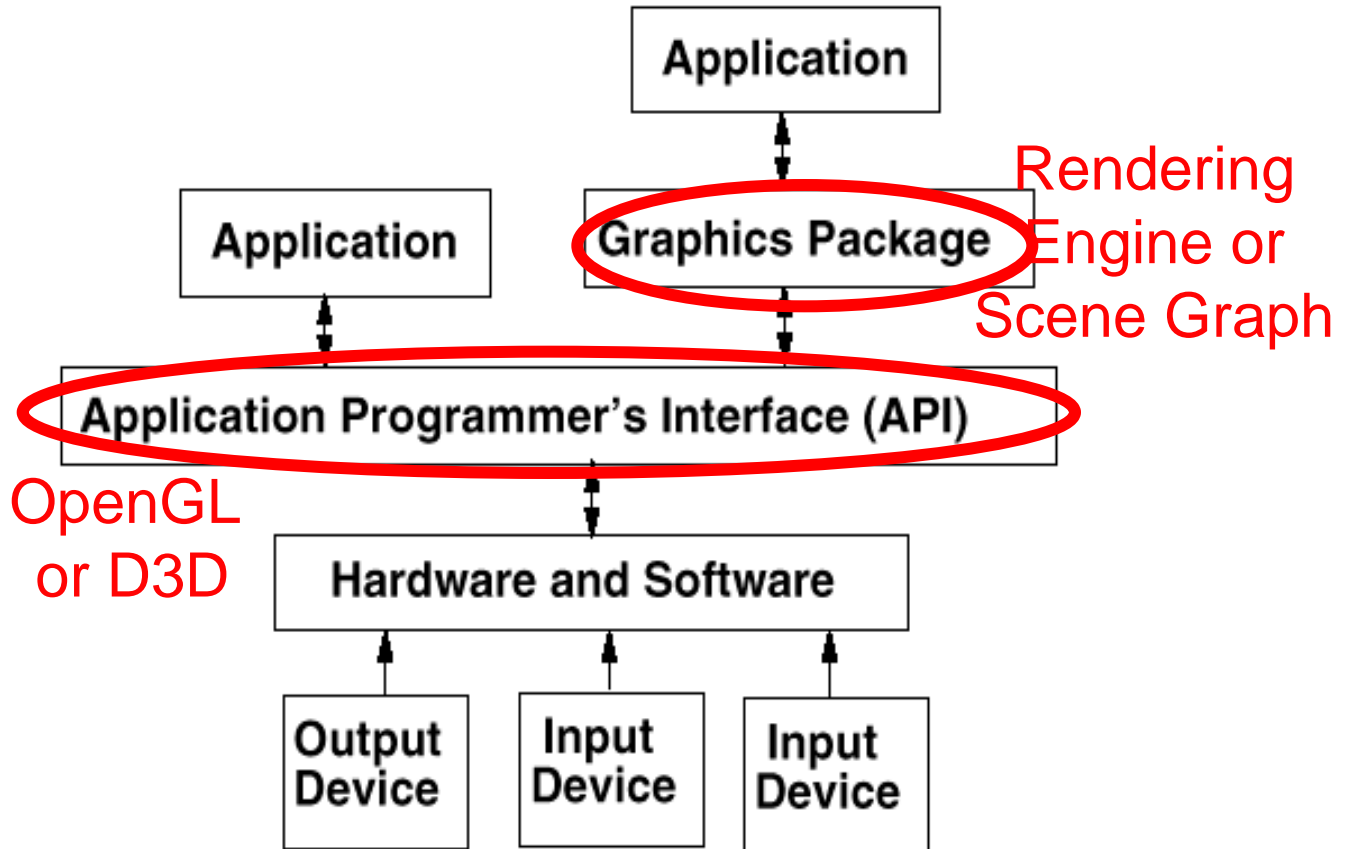
Hardware

- 3D Graphics Hardware
- Input Devices
- Output Devices

Software

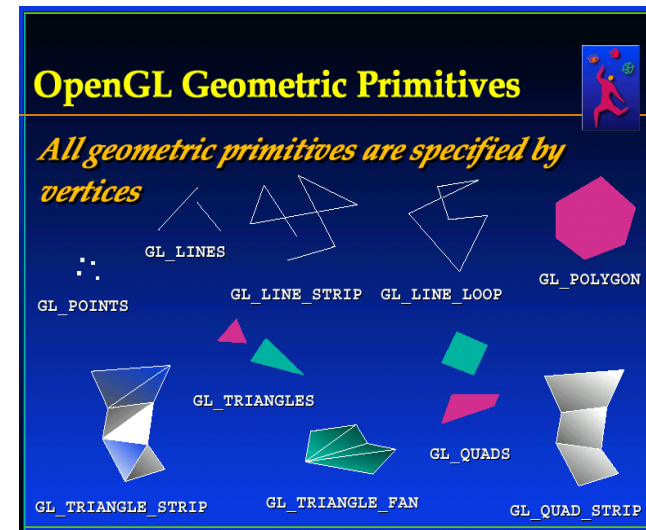
- 3D Interaction
- **High Level Graphics Programming**
- Usability, Evaluations & Psychological Effects

Application Programmer's View



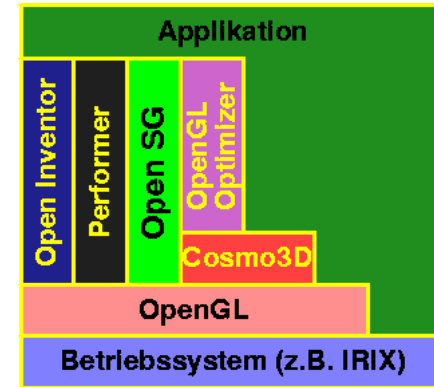
Low-Level Graphics API

- OpenGL (v 1.0 1992), Direct3D (DirectX 2, 1996)
- Procedural
- Primitives
 - Line, Triangle, ...
 - Color, ...
- Dual Database Problem
 - 1. Representation: Data Objects
 - 2. Representation: Graphical
 - Redundancy, Problem of Inconsistency

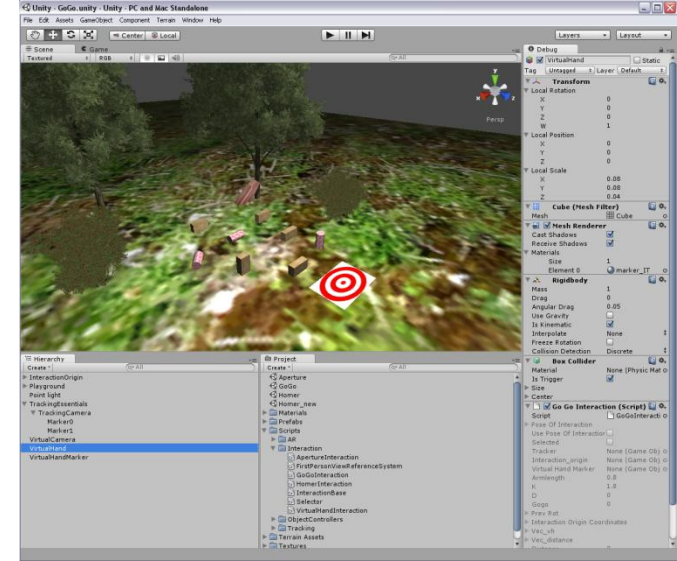


High-Level Graphics API

- e.g. SGI Inventor, VRML, Java3D, OpenSG, Performer, World Toolkit
- Rendering Engine (e.g. Ogre, Unreal Engine,...) or Scene Graph
- Object oriented
- Scene Objects – “Objects, not Drawings”
 - Not limited to graphical display
- Interactivity: Event-model for 3D scenes
- Software architecture
 - Toolkit-approach, extendible



Lab Exercise: „Higher Level Programming“



- Game-Engine

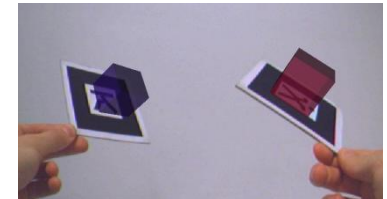
- E.g. Unity3D



- Extended functionality: „Simple AR Framework“

- Tracking input (Qualcomm SDK)
 - Distribution
 - (Stereo rendering)

AR Toolkit Plus
Augmented Reality Tracking Library

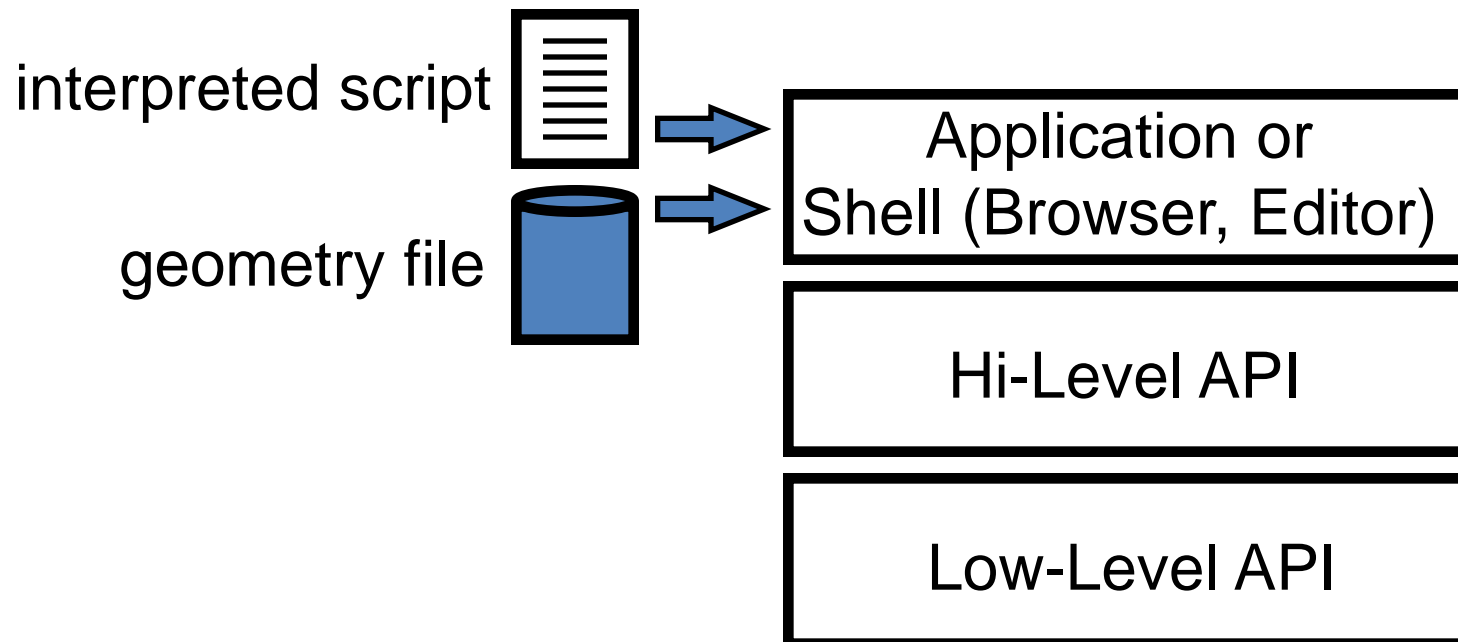


- Object oriented programming in C# / Javascript

- Based on scene graph

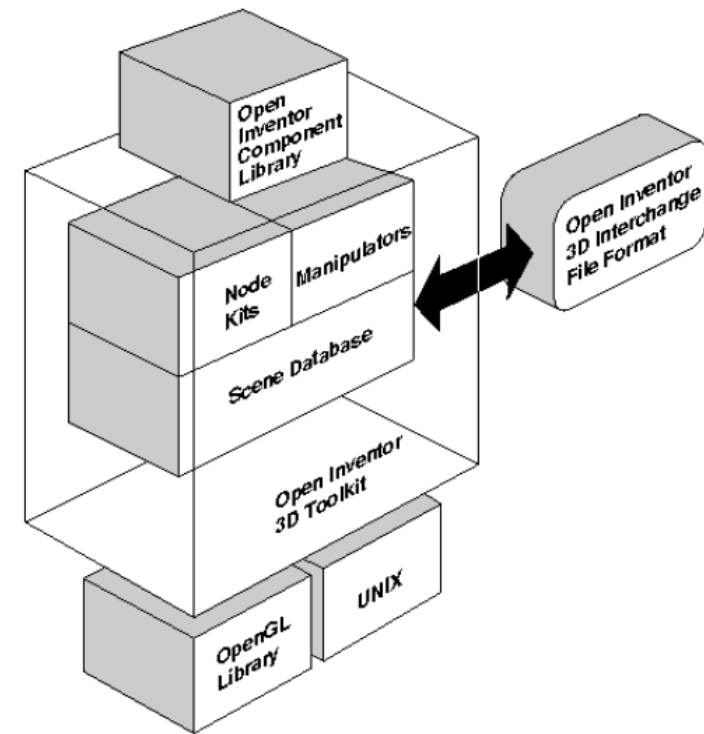
Why High-Level API?

- Rapid prototyping
- Rapid application development (RAD)



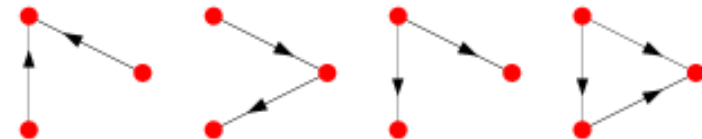
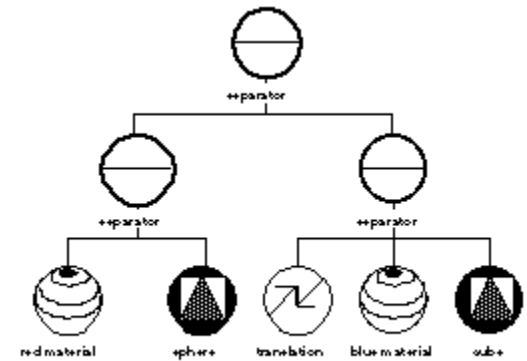
Scene Graph Example: SGI Open Inventor

- Scene graph library
 - Based on C++
 - Used in research & commercial projects
 - Platform & windowing system independent
1. Version: SGI Inventor, 1992. Open Source (ver 2.1)
 2. Version: VGS Open Inventor: Continued development of SGI Inventor. Now ver. 8.0
 3. Version: **Coin** by Systems in Motion (SIM), Re-Engineered API, Open Source; ver. 3.0
<http://www.coin3d.org/>



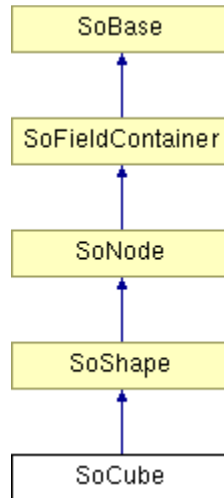
Scenegraph – Structure

- Graphical data structure = **Scenegraph**
- Scenegraph consists of **Nodes**
- Directed graph! (Head/Tail)
Directed edges -> **Hierarchy**
- Use of the hierarchy
 - Semantic Hierarchy: e.g. car (parts)
 - Geometric Hierarchy: e.g. puppet / jointed doll
- Usually one tree is sufficient
- General: **Directed Acyclic Graph** (DAG)
 - [Multiple parent nodes allowed]
 - No directed circles



Scene graph - Nodes

- Nodes consist of data and methods
- Nodes are of a specific **type**
 - Type determines behavior
 - Behavior = Reaction to certain events
 - Events are generated by the application – by the user -> Interactivity
- Nodes are instances of a **class**
 - Scene hierarchy vs. class hierarchy!
- Flexibility: Choose node(type), compose scene graph with nodes
- Extendible: New nodes can be implemented



Scene graph - Fields

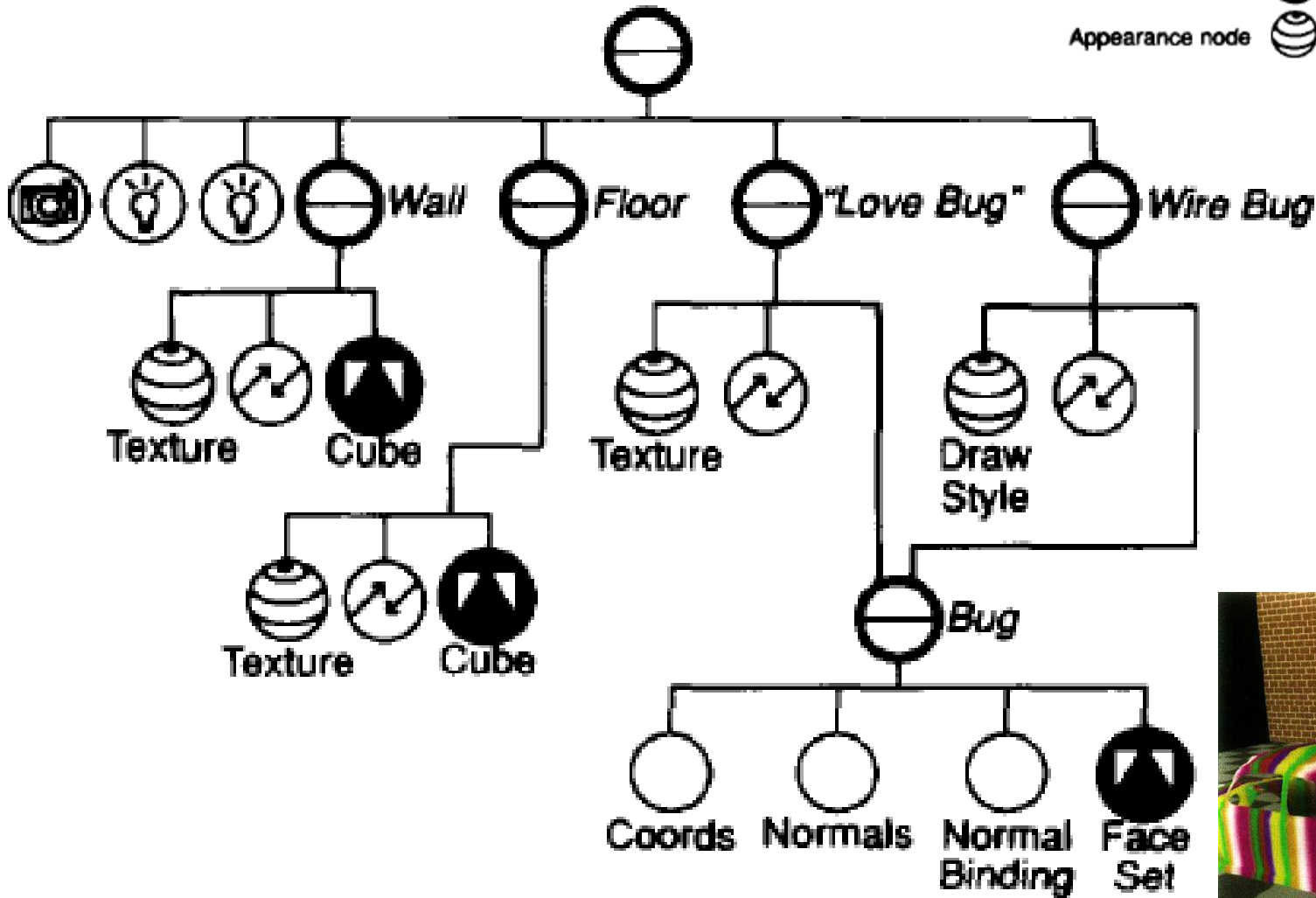
- Attributes (member variables) in nodes are called **fields**
- Fields: set/get, detect changes, connect fields across nodes
- Fields are objects by themselves
 - Float-Object, String-Object etc.

SoMaterial

ambientColor
diffuseColor
specularColor
emissiveColor
shininess
transparency

Example

- Separator group node
- Other group node
- Camera node
- Light node
- Shape node
- Appearance node
- Transform node
- Other property node
- Node Kit
- Manipulator
- Sensor
- Subgraph



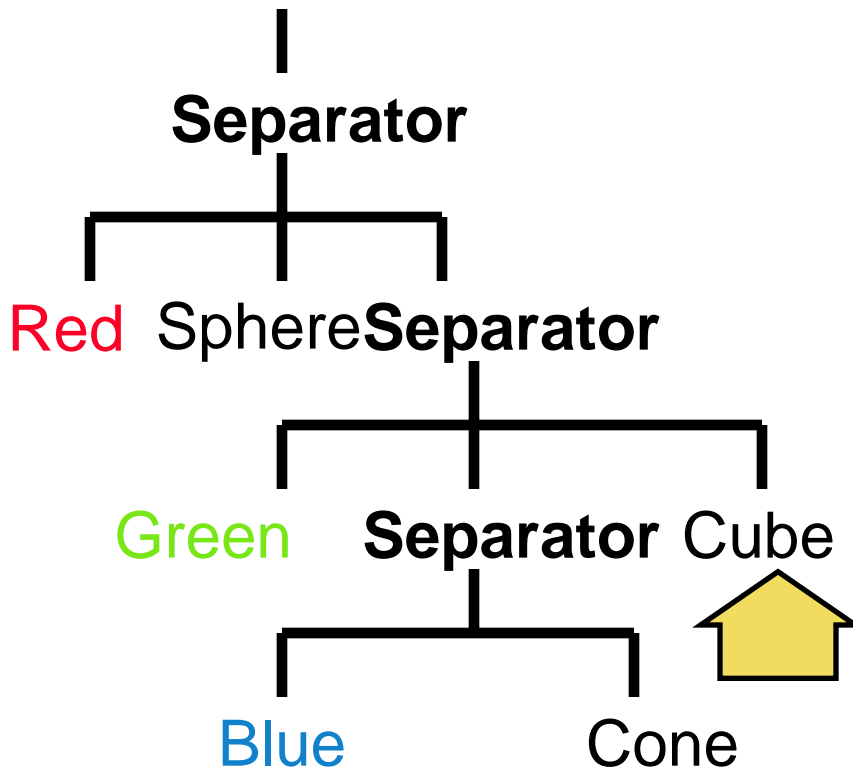
Graph Traversal: Basic Idea

- Data structure (scene graph) is processed (=“traversed”)
- For each node a number of methods is implemented:
 - Rendering
 - BoundingBox calculation
 - Transformation matrix calculation
 - Handle Events (e.g. picking)
 - Search nodes
 - Write to file
 - Execute user callback...

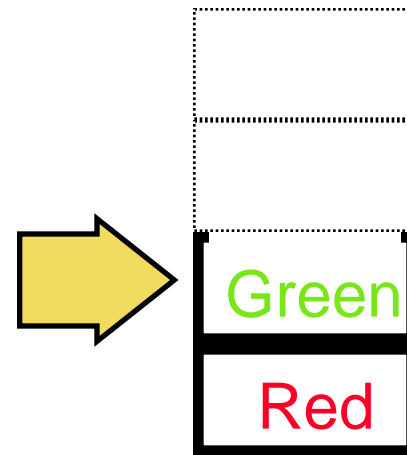
Graph Traversal Order

- All nodes must be processed
- In general: Depth-First
- Traversal uses a State-Engine
- Difference in Group Nodes
 - Ordered Group
 - State Propagation top->down and left->right
 - e.g. Inventor, VRML
 - Very flexible scene graph generation
 - Unordered Group
 - State Propagation only top->down
 - e.g. Java3D, Performer
 - Parallel Render Traversal possible (Threads, SMP)!

State, Stack & Separator



Color state stack

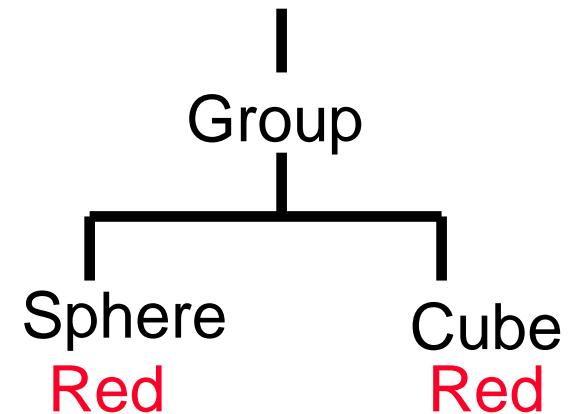
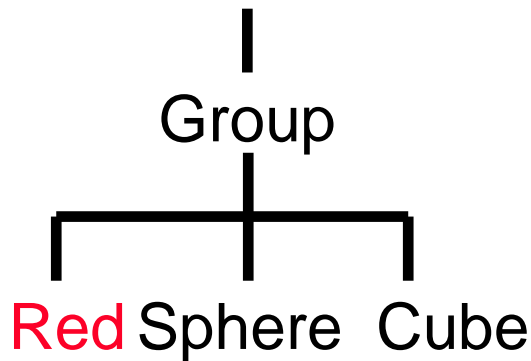
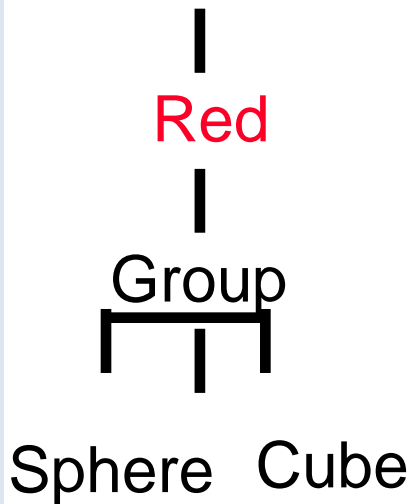


Traversal saves state in Stack

Graph Traversal

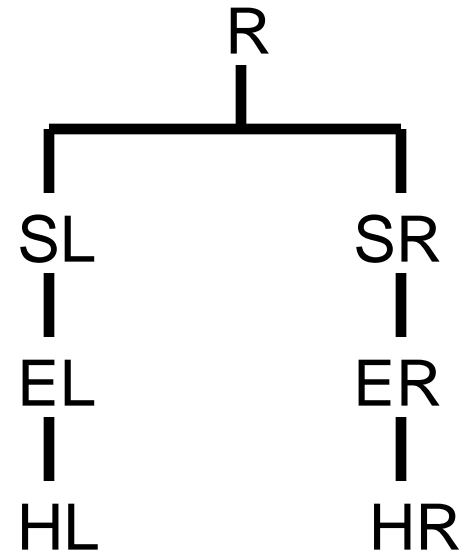
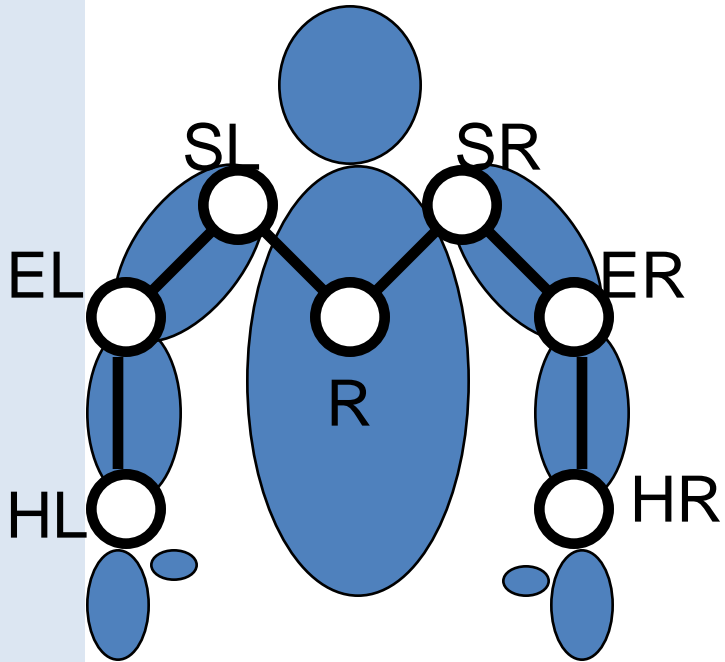
Modeling Attributes

- In-between, leaves or fields?



Some toolkits only allow specific structures
 e.g. VRML97 Shape & Appearance combined

Transformation-Hierarchy



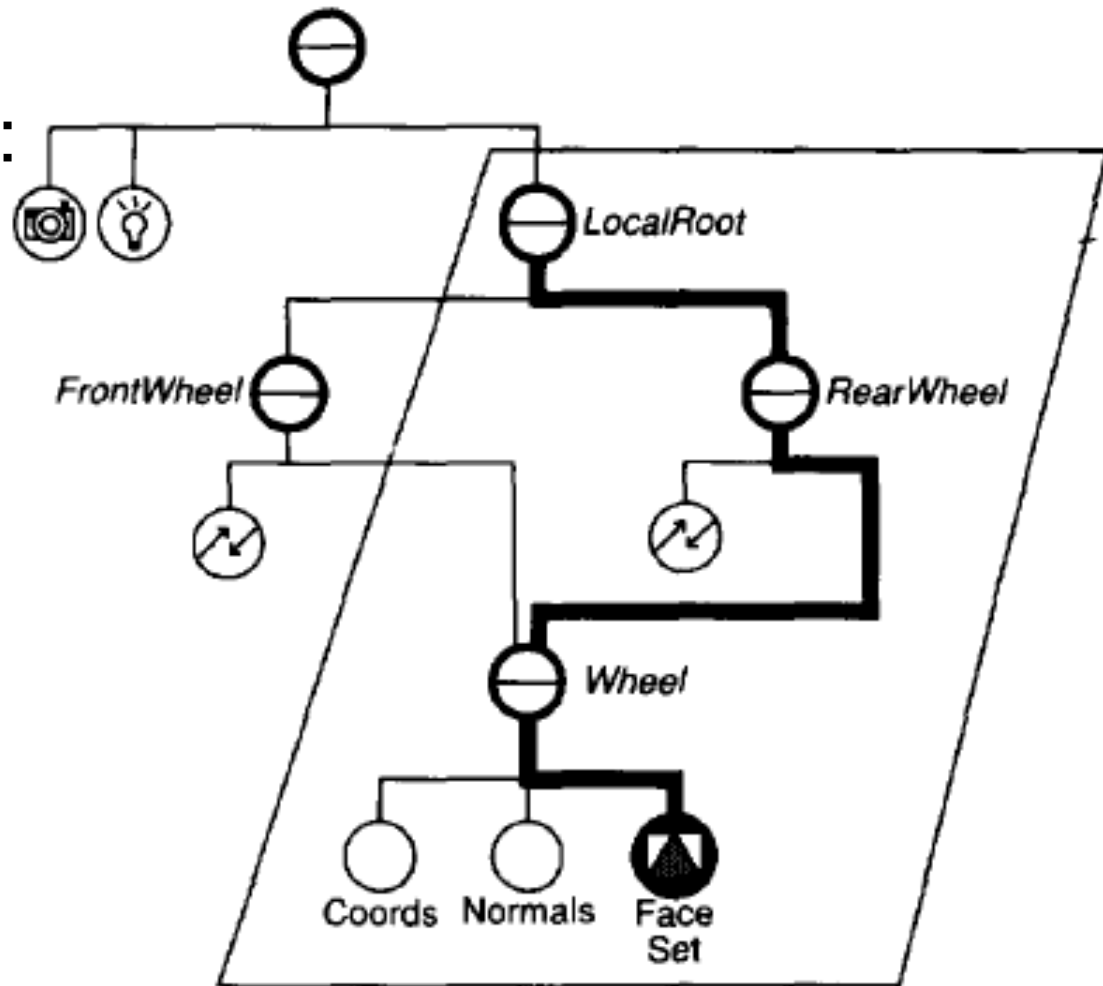
OpenGL Matrix Stack \leftrightarrow Transformation hierarchy

Instancing (Re-use)

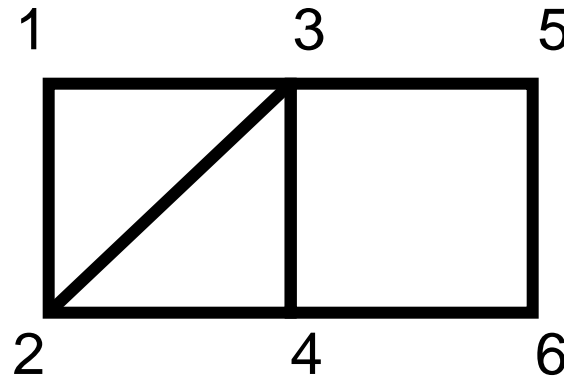
Example: Car

In case of textures:

- Saves texture memory



Polygonal Shapes: Coordinates



Indexed vs. non-indexed polygon lists (e.g. FaceSet):

Non-Indexed:

$$V = \{P1=(x1,y1,z1), P2, P3, P2, P3, P4, P3, P4, P5, P6\}$$

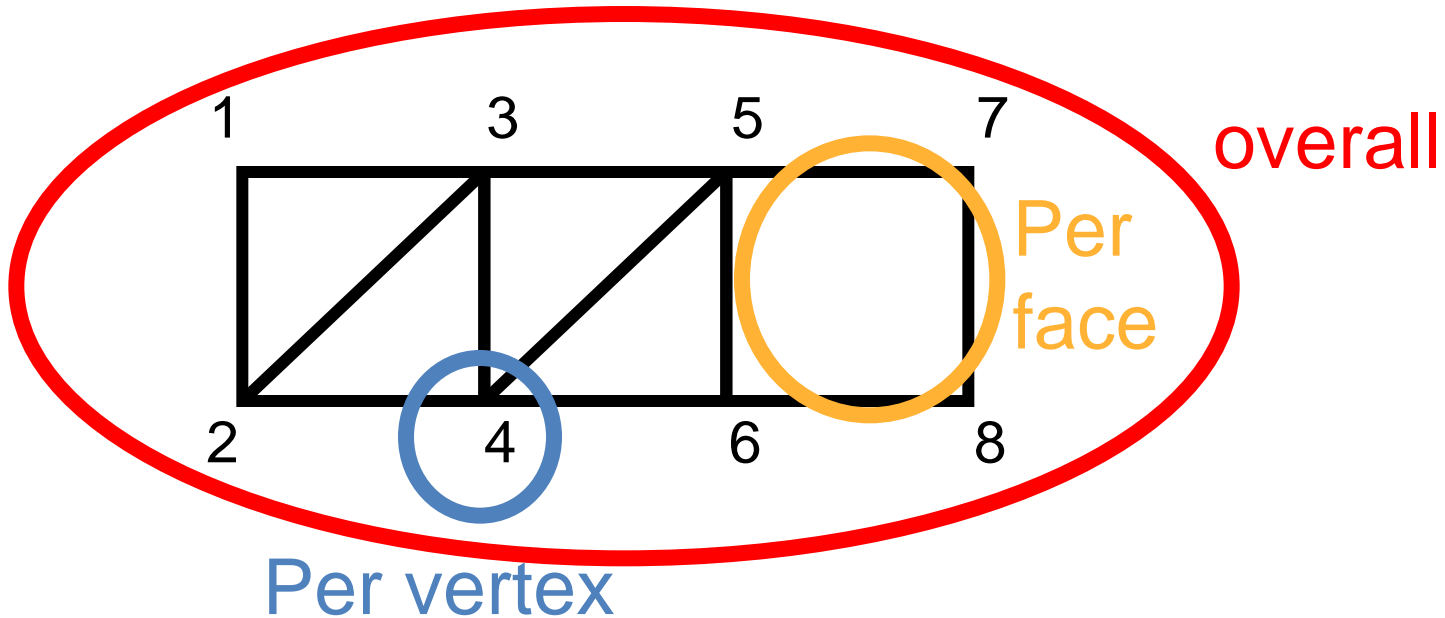
$$F = \{3, 3, 4\}$$

Indexed:

$$V = \{P1, P2, P3, P4, P5, P6\}$$

$$F = \{1, 2, 3, -1, 2, 3, 4, -1, 3, 4, 5, 6, -1\}$$

Polygonal Shapes: Attribute

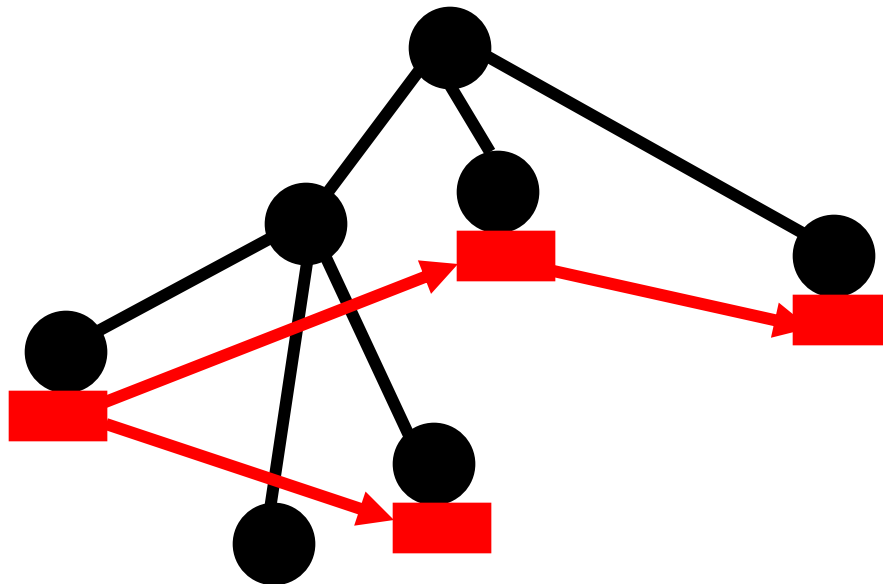


Bindings of attributes

- for material, normals, texture coordinates
- specifies mapping of attributes to polygons
- Overall object, per face, per vertex

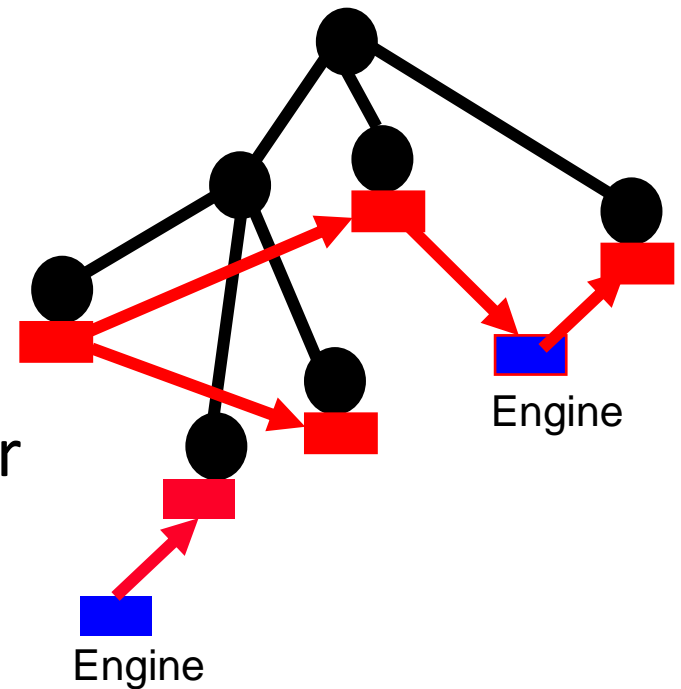
Dependency Graph

- “Field connections”: Field types must be compatible!
- Two different (overlapping) structures
 - Scene graph
 - Dependency graph (dependent fields)



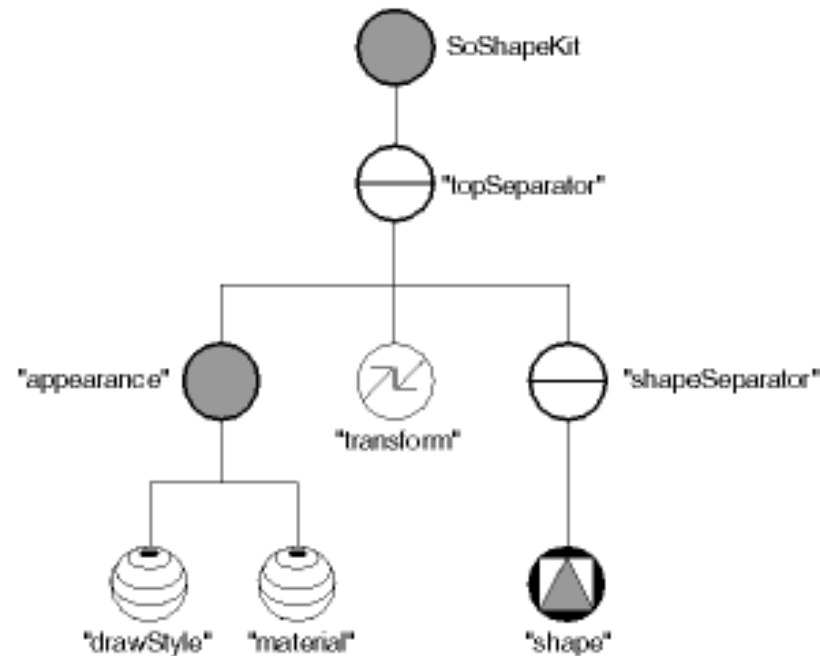
Engines

- To model complex dependencies in graph
- TargetField := Engine(SourceField)
- E.g. Calculator, Counter, Type converter, Interpolator, Trigger



Node Kits / Prefabs

- Prefabricated sub-scene graphs
 - e.g. transformation, material + shape
 - Simplify the construction of semantically correct scenes

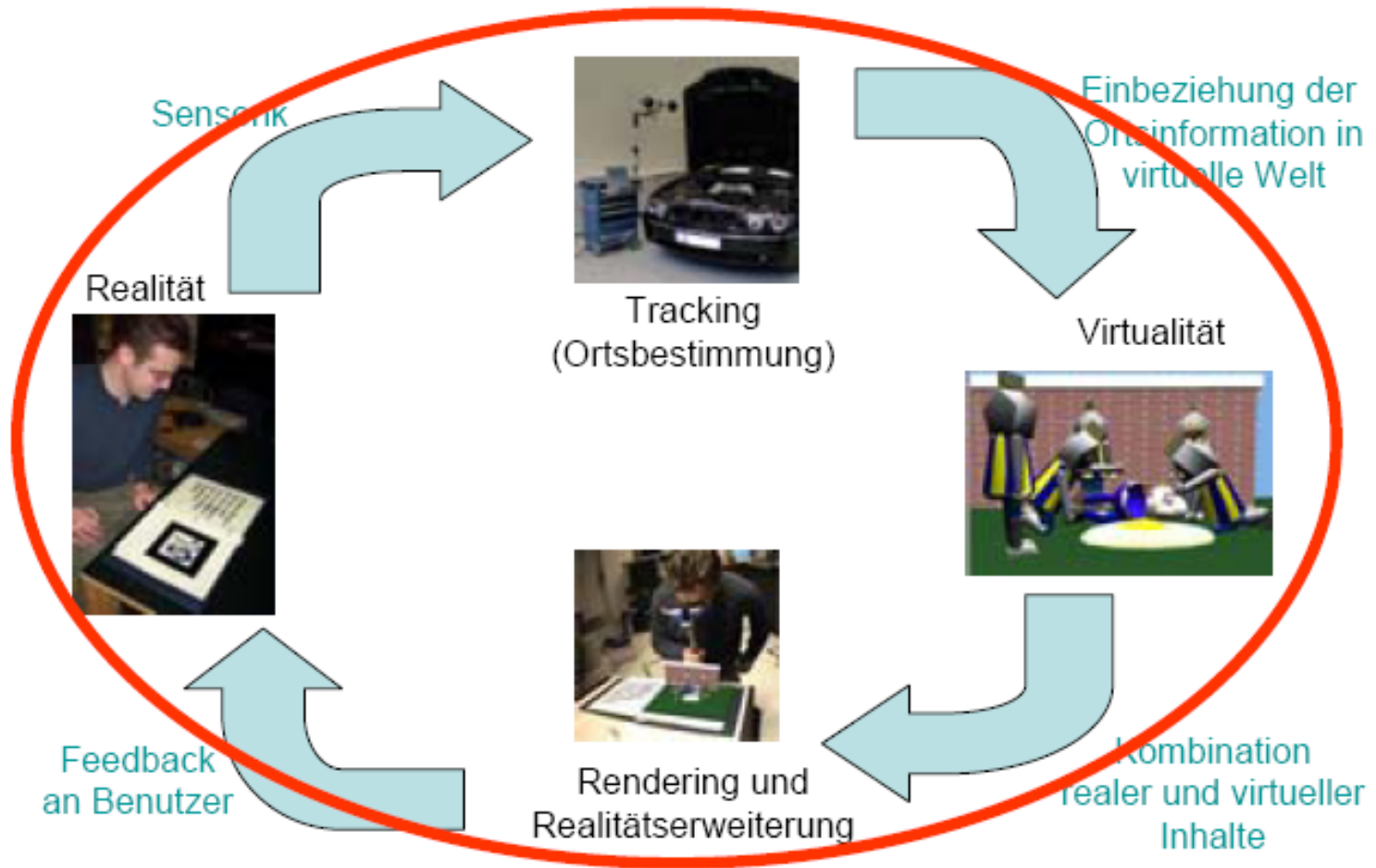


Software Design and Components of an VR/AR Framework

Hannes Kaufmann

Interactive Media Systems Group (IMS)
Institute of Software Technology and
Interactive Systems

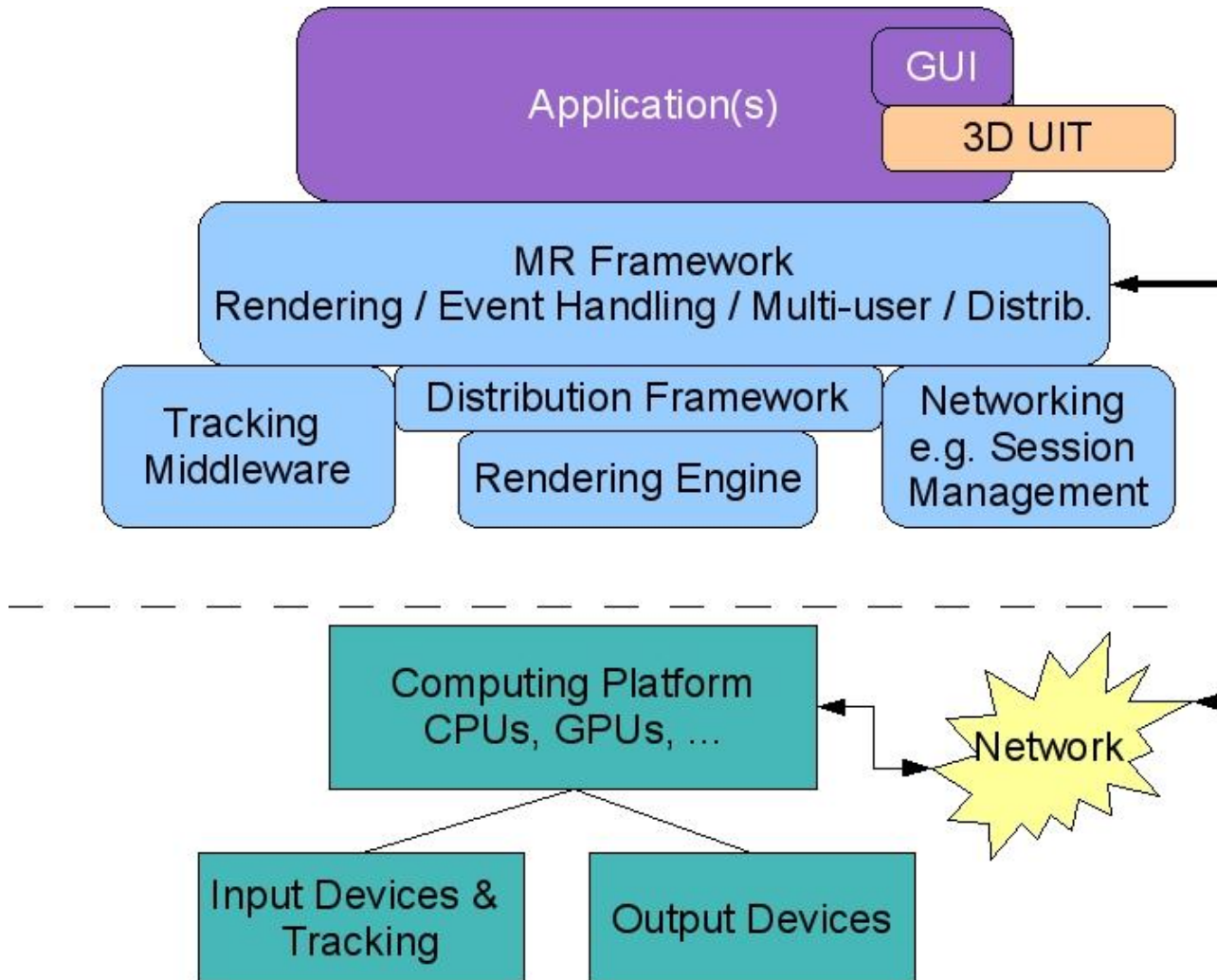
Ein Generisches AR-System



AR/VR Framework: Requirements & Wishes

- Support multiple input & output devices
 - Input: Interface to tracking middleware (e.g. OpenTracker, VRPN)
 - Output: High level graphics programming, Stereo rendering,...
- Handle user interaction
- Allow flexible 3D user interface
 - widget libraries/middleware
- Support of collaboration
 - multiple users, flexible user configuration, mobile work
- Support distributed work
- Easy application design / authoring

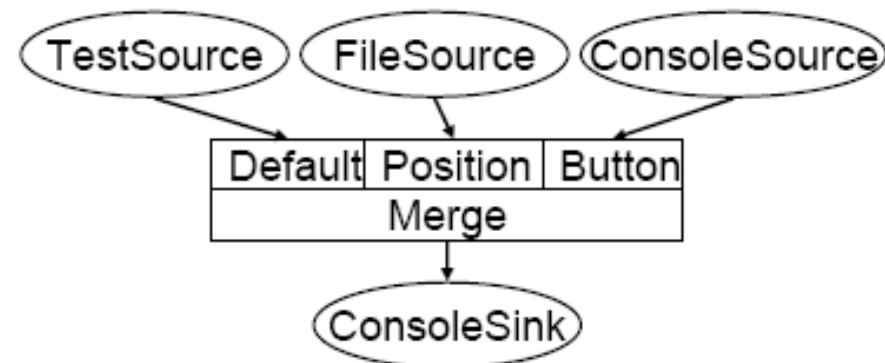
VR/AR/MR System Architecture



OpenTracker - Intro

- Software library - Interaction framework
- Implemented as data flow network
- Provides drivers for multiple input devices
- Transformation & Filtering of tracking data
- Configuration with XML files
- Flexible distribution of tracking data
- Changes of configuration at run time possible with OT 2.0

www.studierstube.org/opentracker



Example AR Framework: STUDIERSTUBE

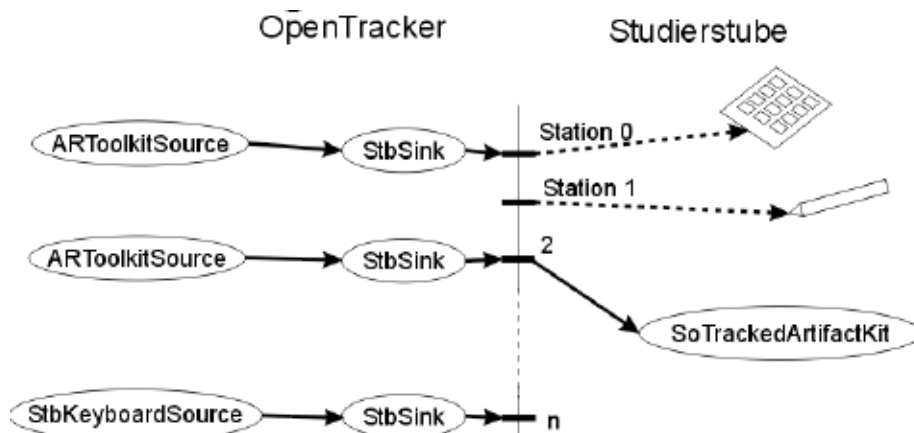
- “*study room*”
- A long term research project to develop a powerful **collaborative Augmented Reality** platform
- 1996-today



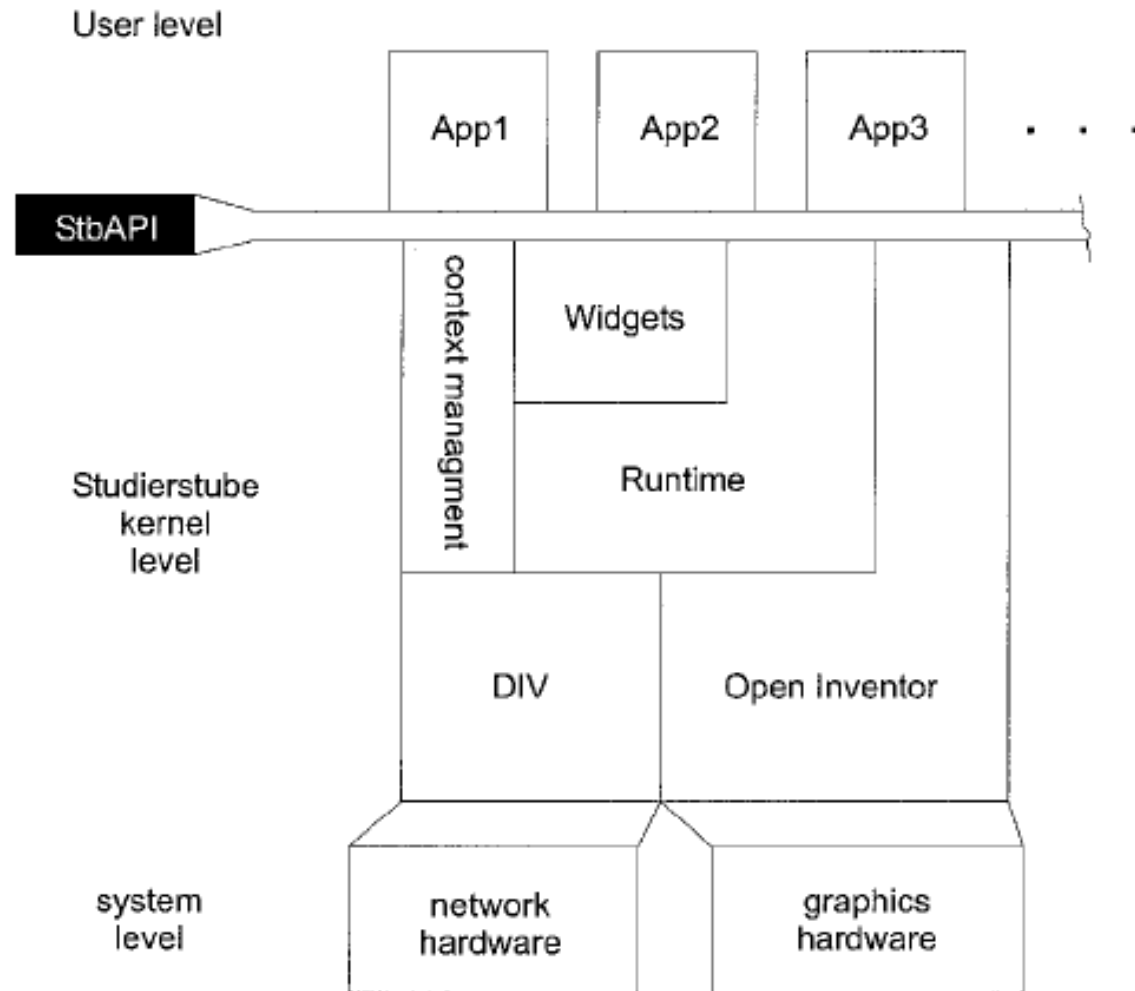
Faust and Mephisto
in the study room

Studierstube Introduction

- Principle: scene graph library
- Basic idea:
 - Rendering is an integral part of an AR application
- **Application logic:**
 - is encapsulated **in nodes** of the scene graph.
 - automatically executed at traversal of the scene graph
- Integration of OpenTracker framework



Studierstube Architecture

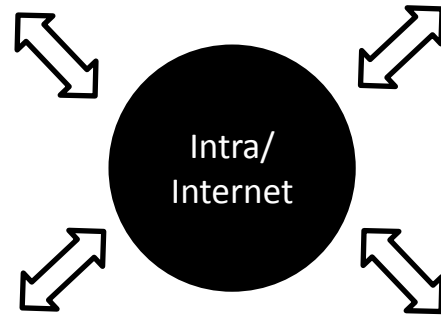


Studierstube

- Studierstube Kernel extends Open Inventor scene graph
 - 3D event system (6DOF)
 - 3D Widgets
 - Applications in 3D Windows (dynamic loading) -> Multitasking supported
 - Multi-user support
 - **Distributed Open Inventor (DIV)** – replicate scene graph on multiple machines
 - Multicast vs. TCP/IP network protocol
 - Robust application replication

Example: Distributed VR / AR in Education

- Distributed collaboration over large distances
- Large number of users supported
- Flexible hardware setups
- Interaction depends on input device properties

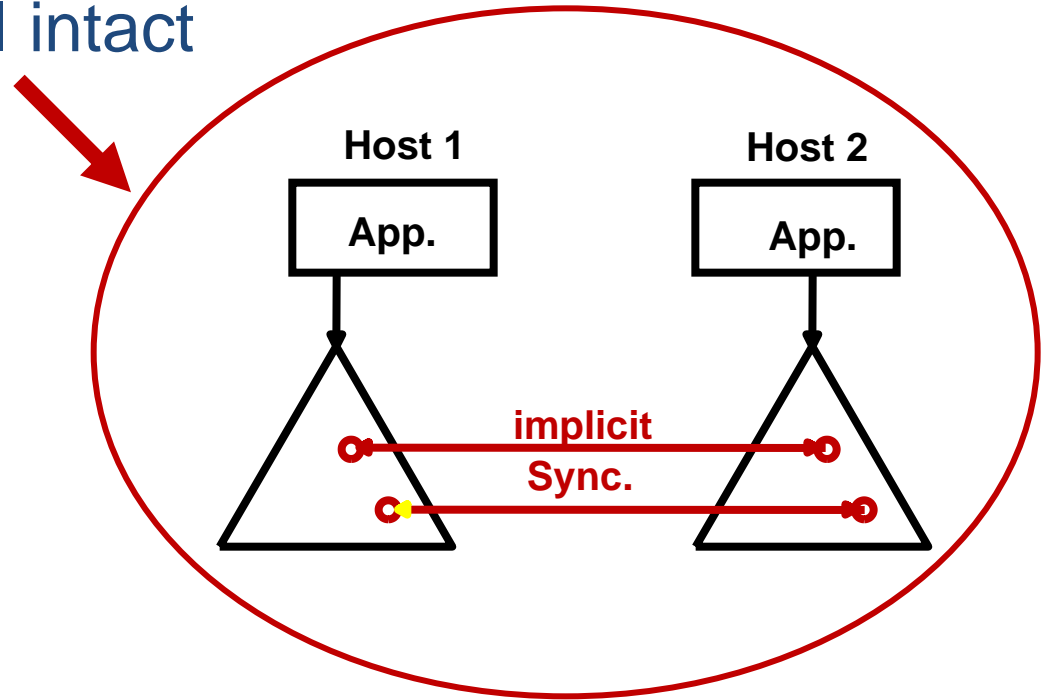
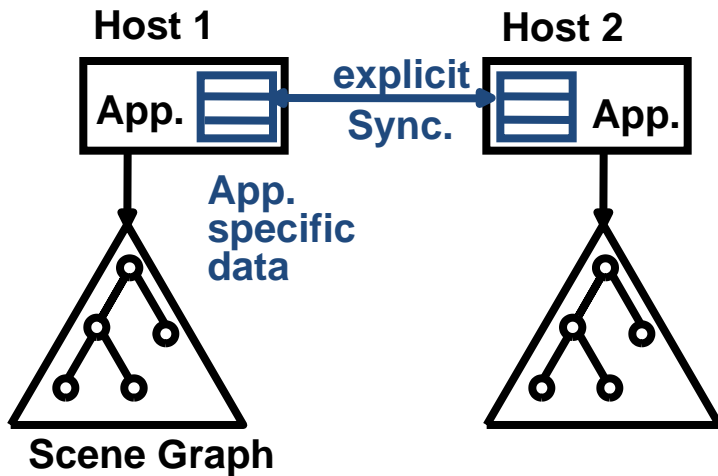


Distributed Shared Scene Graph

- Shared Memory (SM): Multiple CPUs access the same memory
 - Very simple and popular
 - May need mutual exclusion (locks etc.)
- Distributed Shared Memory (DSM):
 - SM on top of standard message passing
- Distributed Shared Scene Graph: DSM semantics added to a scene graph library

Symmetric Approach: Distributed Shared Scene Graph

Goal: Distribution without programming
Keep existing API intact



- Dual database (app, scene)
- Optimizations

- Distributed shared memory semantics
- Transparent distribution
- E.g.: COTERIE, Tinmith, Avango, Distributed Inventor (**DIV**)

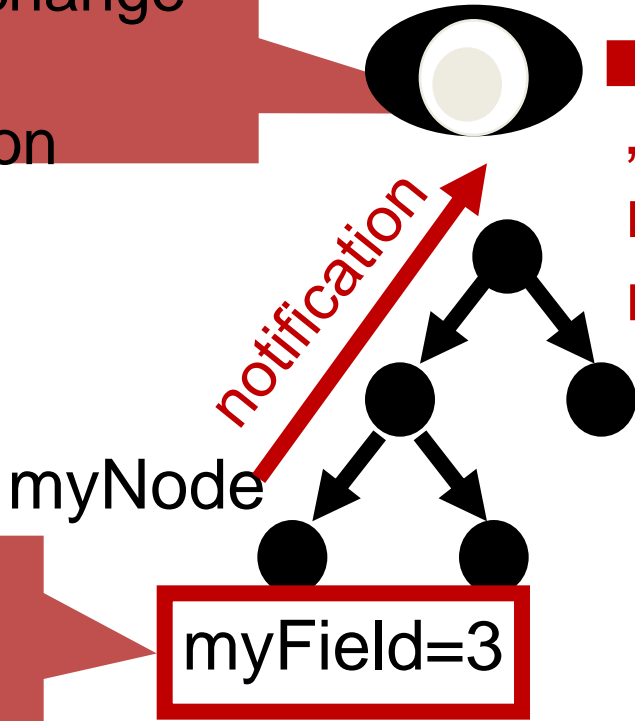
Updates in DIV

2. Observer detects change due to notification

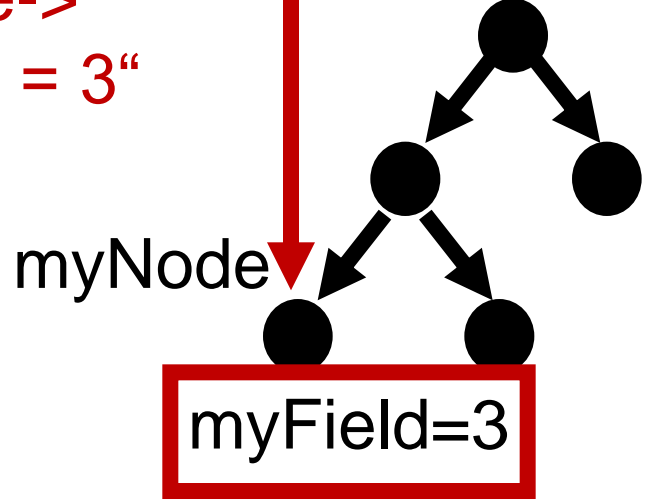
1. App makes update to myField

3. Network transmission of update

4. Receiver applies update

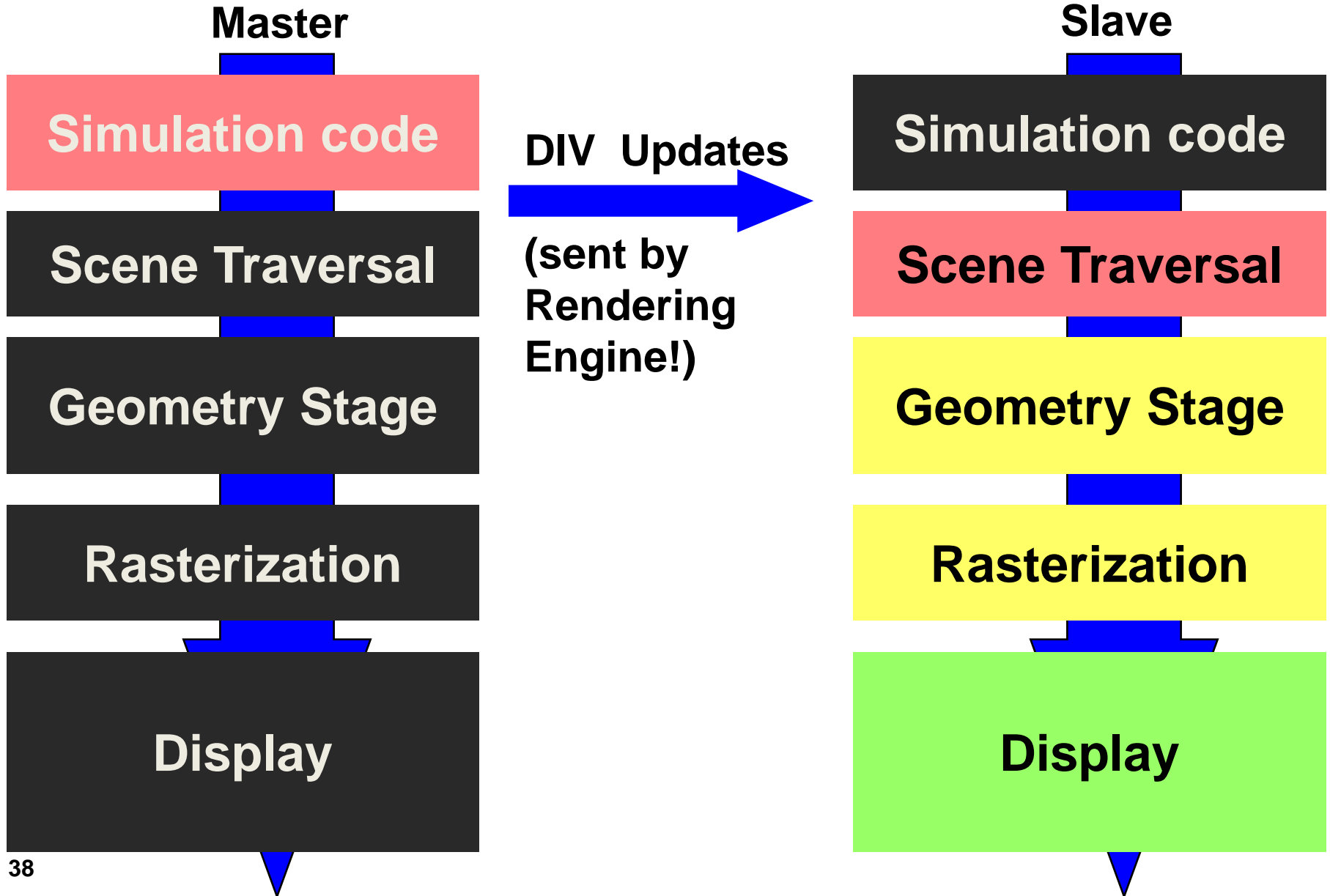


„Update: myNode-> myField = 3“



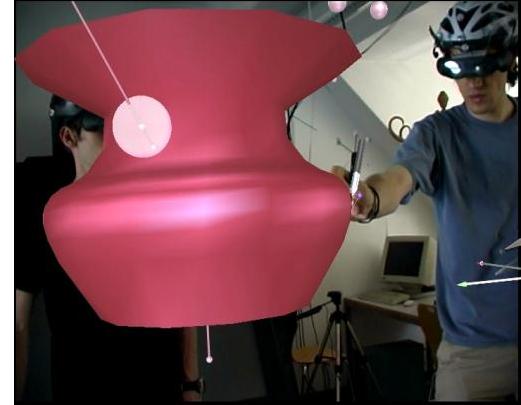
Synchronisation protocol:
- update field
- create node
- delete node
+ some more for for convenience...

DIV - Pipeline



Long Distance Distribution Requirements for AR Applications

- Main Challenges:
 - Robust application replication
 - Reliable network communication over long distances:
 - Networking Protocols (uni-/multicast) & Bandwidth considerations
- 3 Options:
 - **Input data**: e.g. Tracking data of input devices
 - **Output data**: e.g. Application content, Screen
 - **Intermediate data**: High level metadata for regenerating correct application state



Long Distance Distribution - Example

3 Types of Data:

- Input data: e.g. Tracking data of input devices
 - **Tracking Middleware** (e.g. OpenTracker, VRPN)
 - For long distance: Use Unicast (UDP) instead of Multicast
- Output data / Application content
 - **Distributed Open Inventor** (reliable TCP)
- Intermediate data: High level metadata for regenerating correct application state
 - **Construct3D**: enhanced replication behavior
 - Geometric objects not transmitted! Only high level state data

Example: Distribution - Results

- Platform independent (Windows, Linux)
- Long distance (Vienna - Graz)
- 2-6 machines, 5 app. instances
- Dynamic joining & leaving
- Hybrid networks possible (multicast UDP + TCP)
- Educational evaluation
6 students (Sir Karl Popper school)

