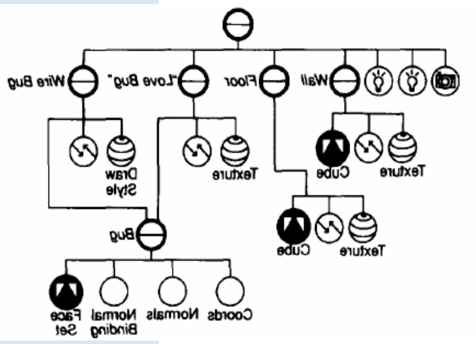
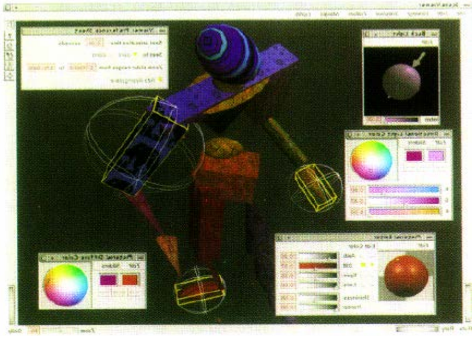
High Level Graphics Programming & VR System Architecture



Hannes Kaufmann



Interactive Media Systems Group (IMS)

Institute of Software Technology and Interactive Systems

Based on material by Dieter Schmalstieg

VR & AR Course Overview

- Introduction

Hardware

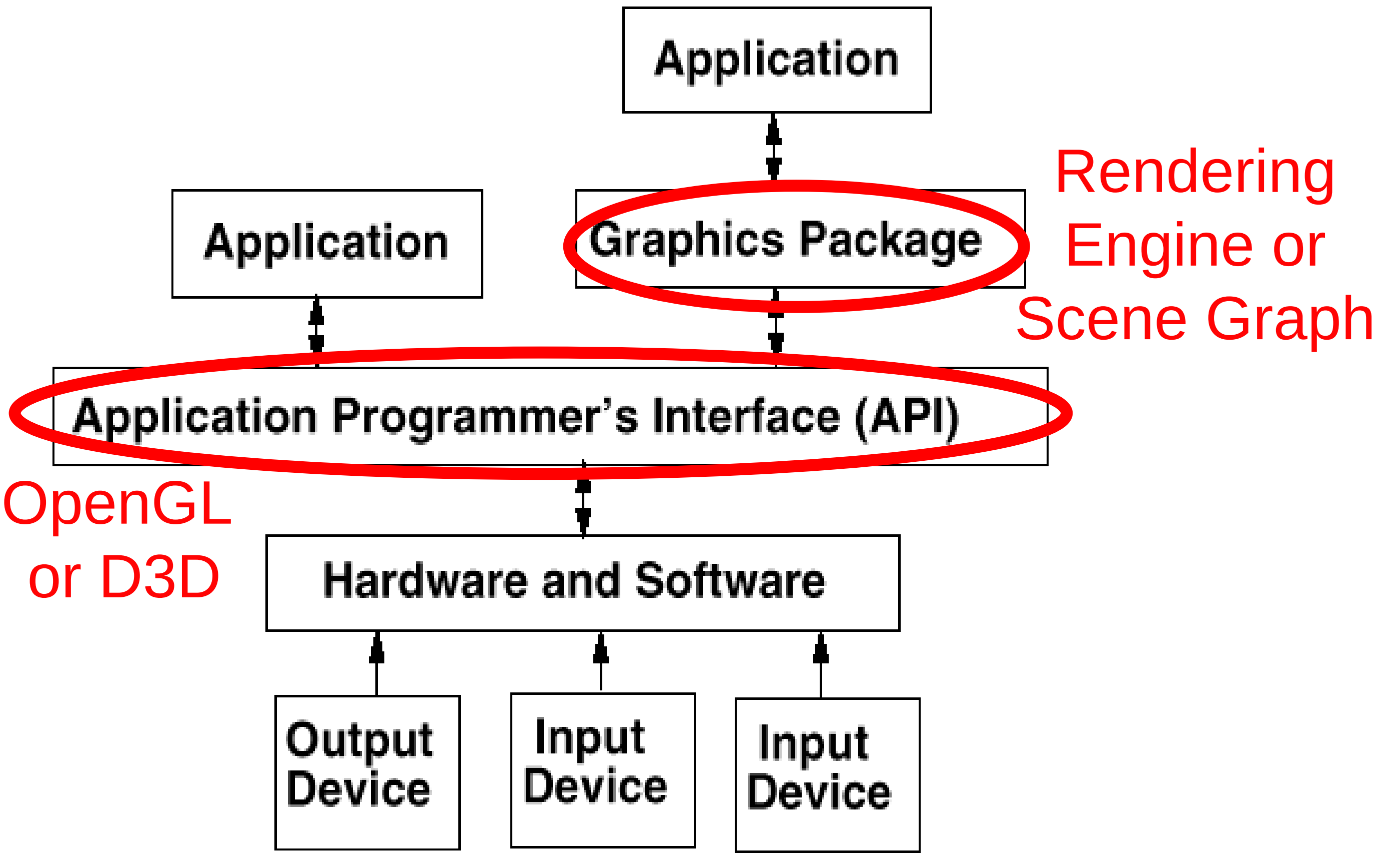
- Input Devices
- Output Devices
- 3D Graphics Hardware

Software

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

Hannes
Kaufmann

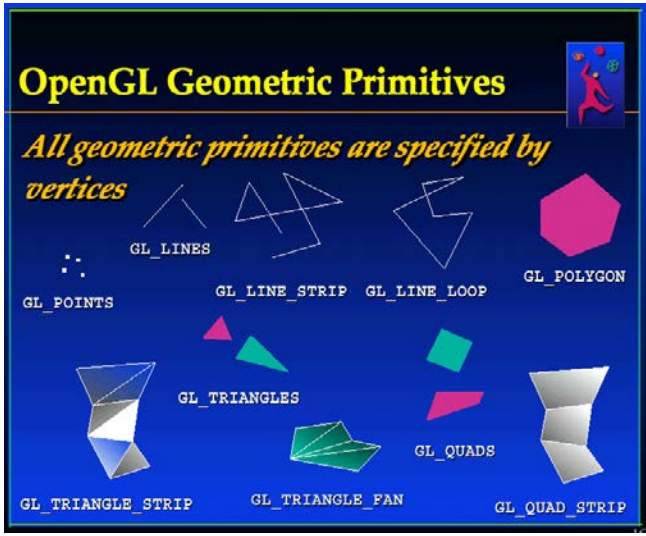
Application Programmer's View



Hannes Kaufmann

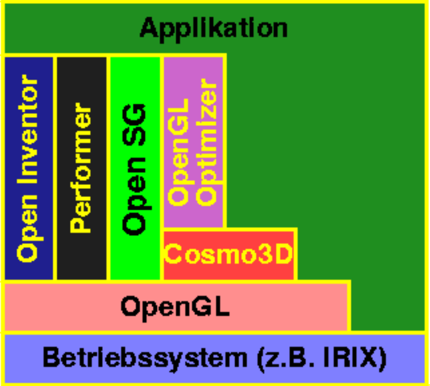
Low-Level Graphics API

- 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



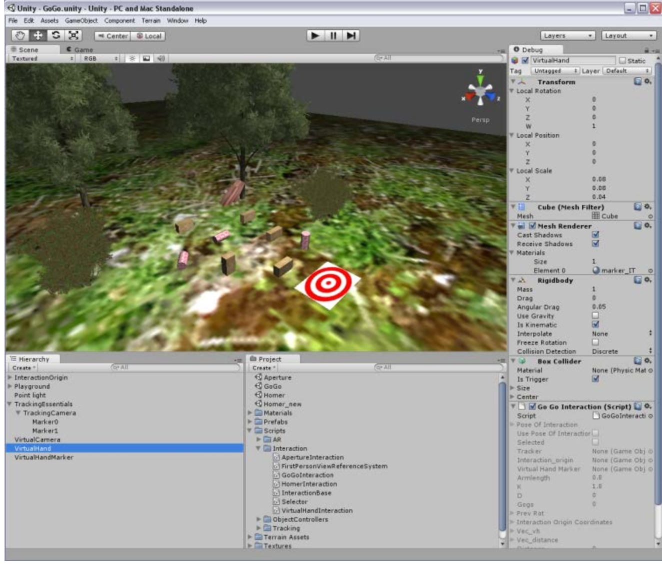
Hannes
Kaufmann

High-Level Graphics API



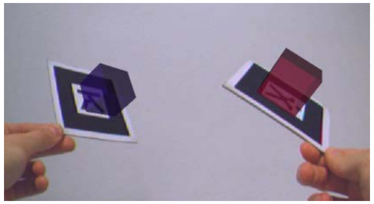
- Rendering Engine (e.g. Unity, Unreal Engine,…) or Scene Graph e.g. OpenSceneGraph, OpenSG, X3D (VRML), Java3D,..
- Object oriented
- Scene Objects – "Objects, not Drawings"
  - Not limited to graphical display
- Interactivity: Event-model for 3D scenes
- Software architecture
  - Toolkit-approach, extendible

Hannes
Kaufmann



Lab Exercise: „Higher Level Programming“

- Game-Engine
  - E.g. Unity3D
  - Extended functionality: „Simple AR Framework“
    - Tracking input
    - Distribution

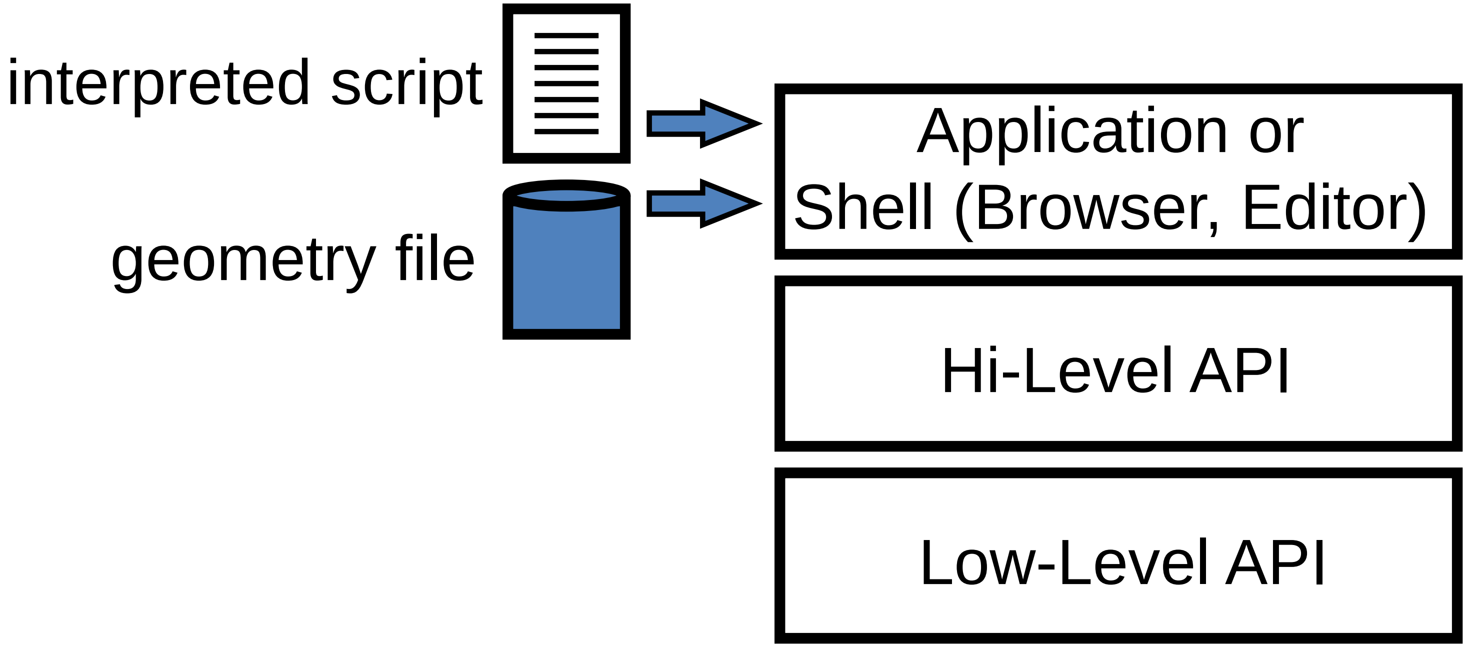


  - Object oriented programming in C# / Javascript
  - Based on scene graph

Hannes Kaufmann

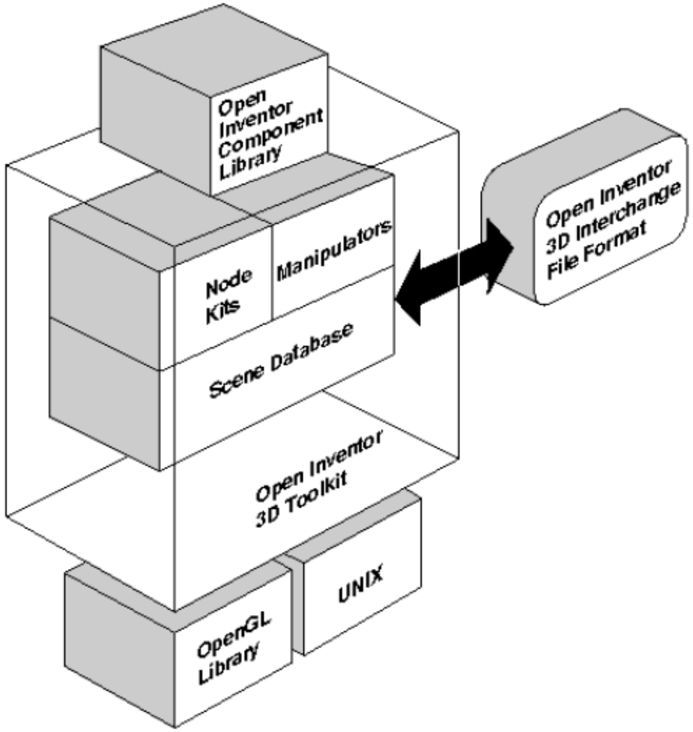
Why High-Level API?

- Rapid prototyping
- Rapid application development (RAD)



Hannes
Kaufmann

Scene Graph Example: SGI Open Inventor



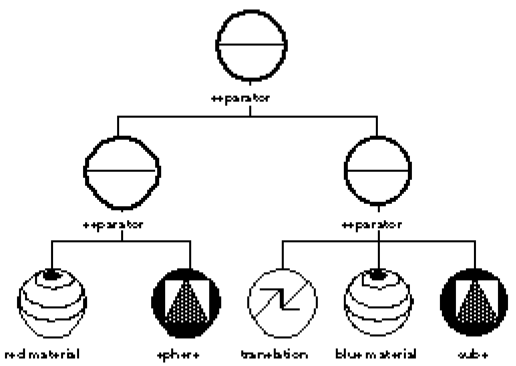
- Scene graph library
- Based on C++
- Used in research & commercial projects
- Platform & windowing system independent

- 1. Version: SGI Inventor, 1992. Open Source (ver 2.1)
- 2. Version: VGS Open Inventor: Continued development of SGI Inventor.
- 3. Version: Coin by Systems in Motion (SIM), Re-Engineered API, Open Source; ver. 3.0
http://www.coin3d.org/

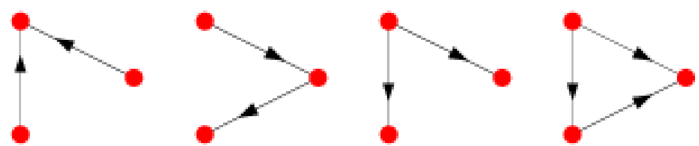
Hannes
Kaufmann

Scenegraph – Structure

- Graphical data structure = Scenegraph
- Scenegraph consists of Nodes
- Directed graph! (Head/Tail)
  Directed edges -> Hierarchy



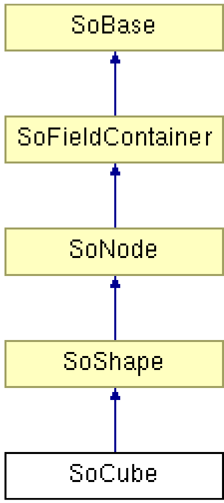
- Use of the hierarchy
  - Semantic Hierarchy: e.g. car (parts)
  - Geometric Hierarchy: e.g. puppet / jointed doll
- Usually one tree is sufficient
- General: Directed Acyclic Graph (DAG)
  - [Multiple parent nodes allowed]
  - No directed circles



Hannes
Kaufmann

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



Hannes Kaufmann

Scene Graph - Fields

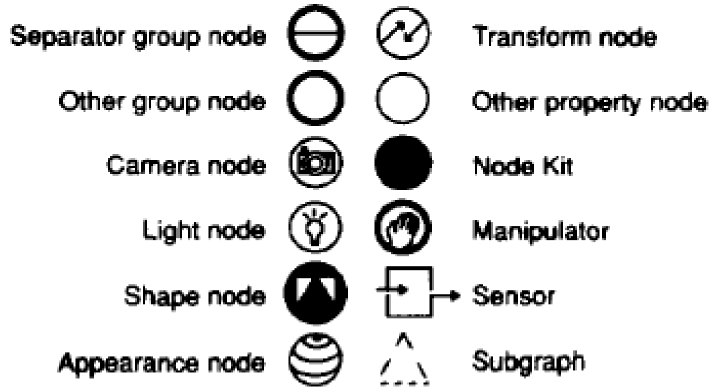
SoMaterial

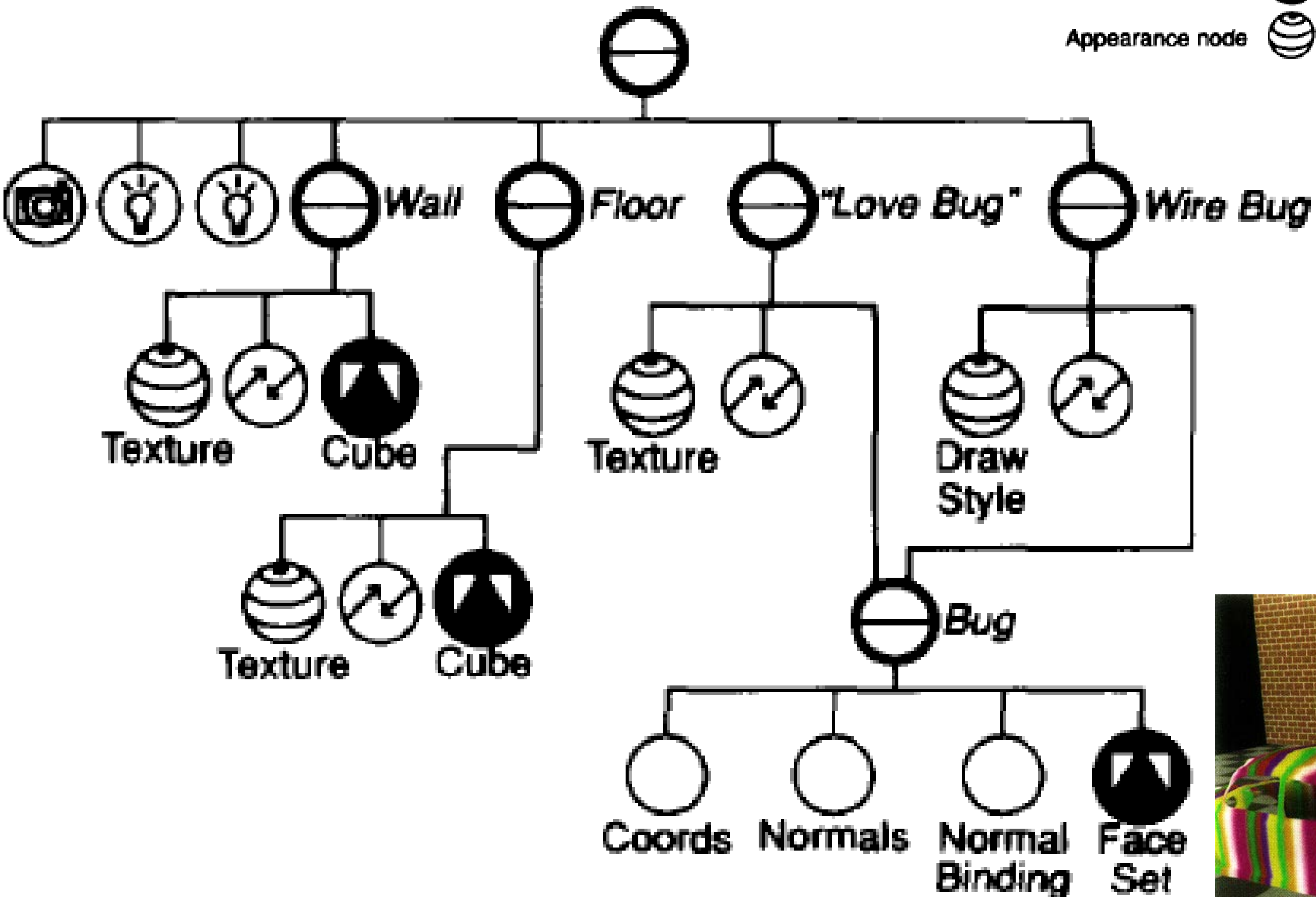
ambientColor
diffuseColor
specularColor
emissiveColor
shininess
transparency

- 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.

Hannes
Kaufmann

Example





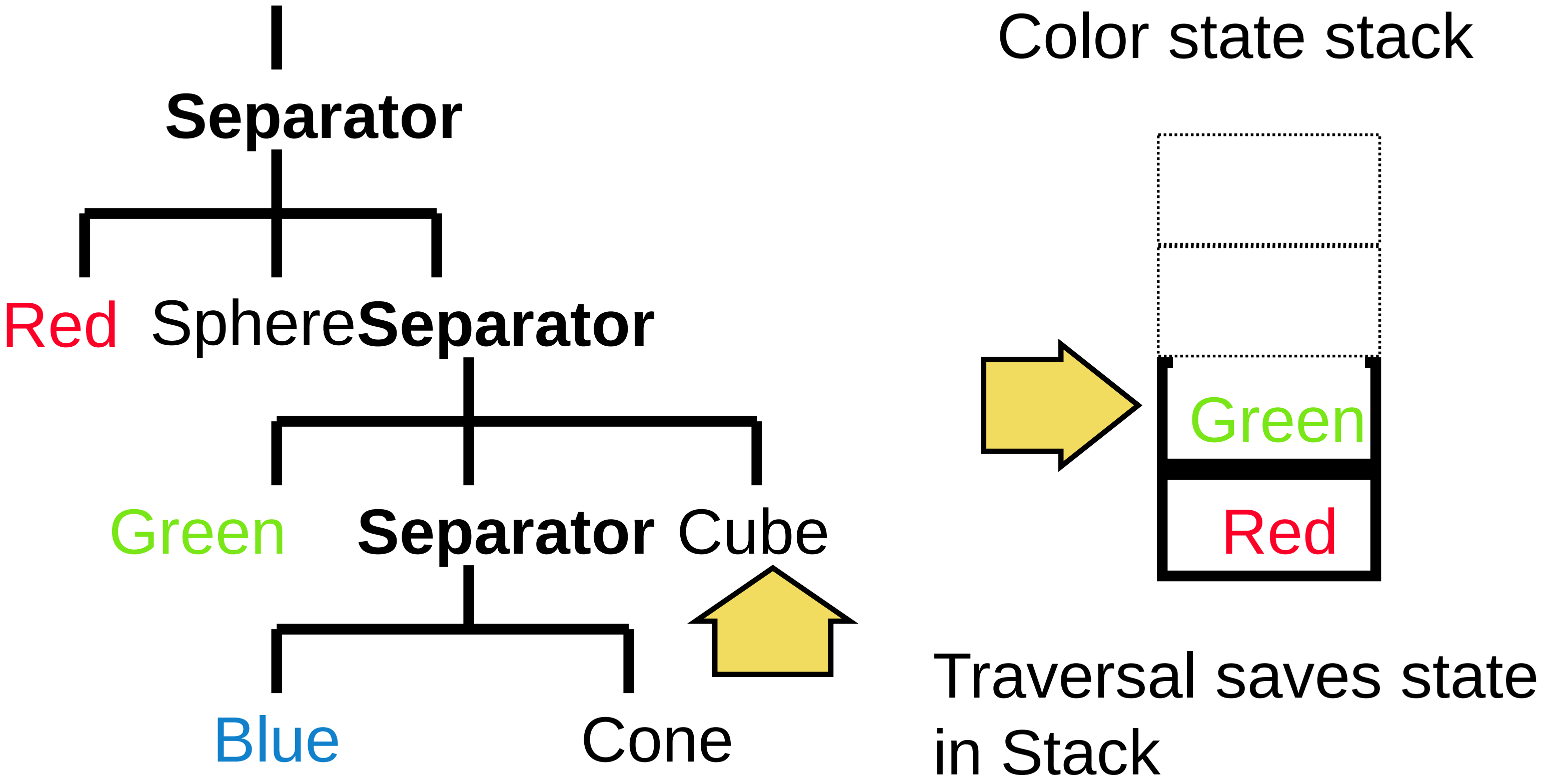
Graph Traversal: Basic Idea

- 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 / X3D
    - Very flexible scene graph generation
  - Unordered Group
    - State Propagation only top->down
    - e.g. Java3D
    - Parallel Render Traversal possible (Threads, SMP)!

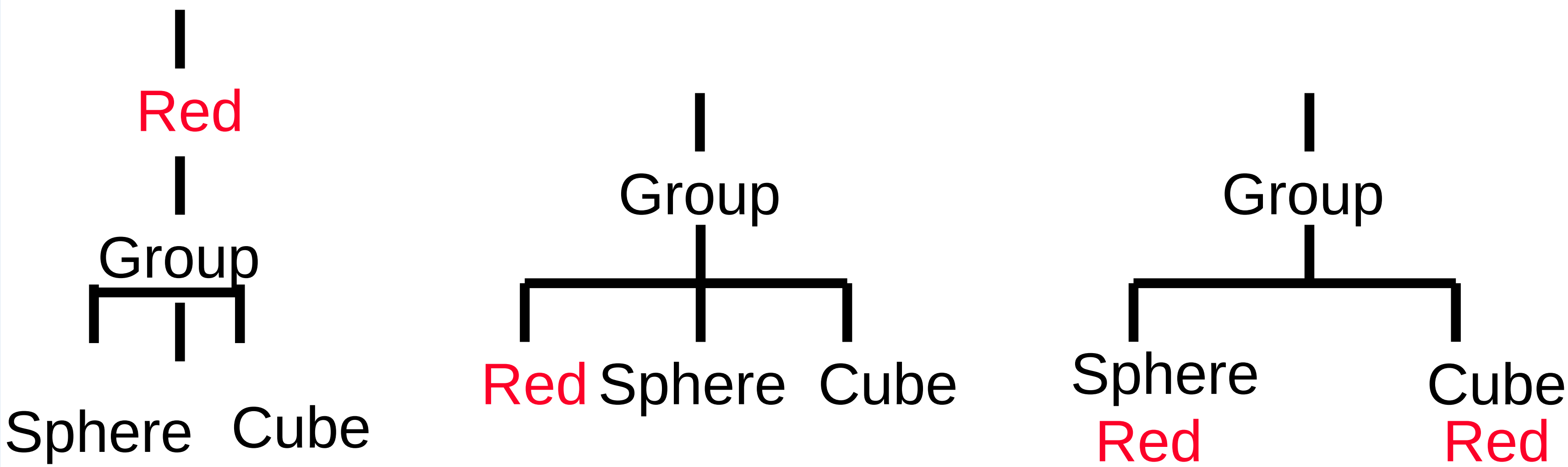
State, Stack & Separator



Hannes Kaufmann

Graph Traversal Modeling Attributes

- In-between, leaves or fields?

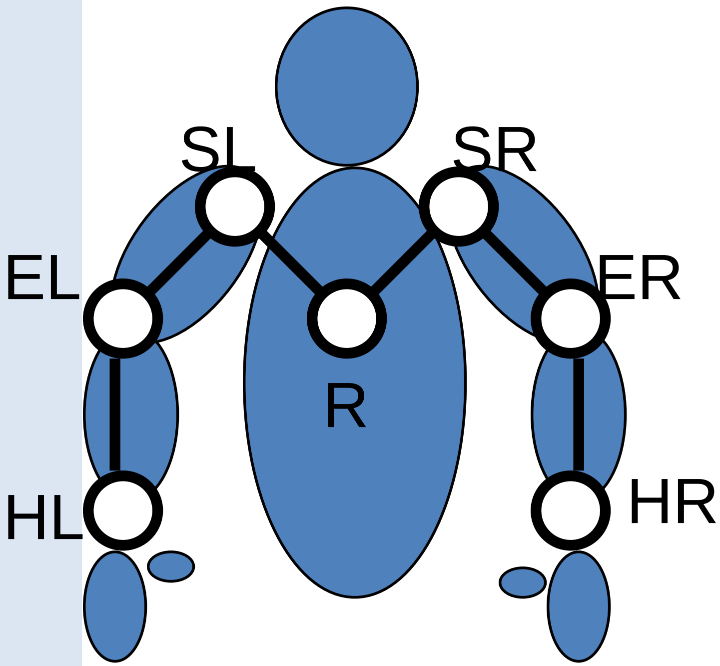


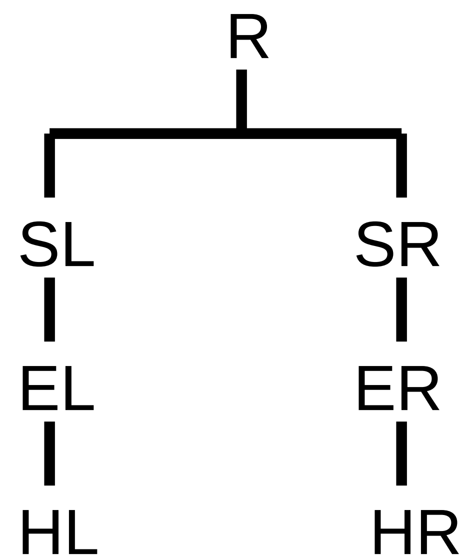
Some toolkits only allow specific structures

e.g. X3D Shape & Appearance combined

Hannes
Kaufmann

Transformation-Hierarchy





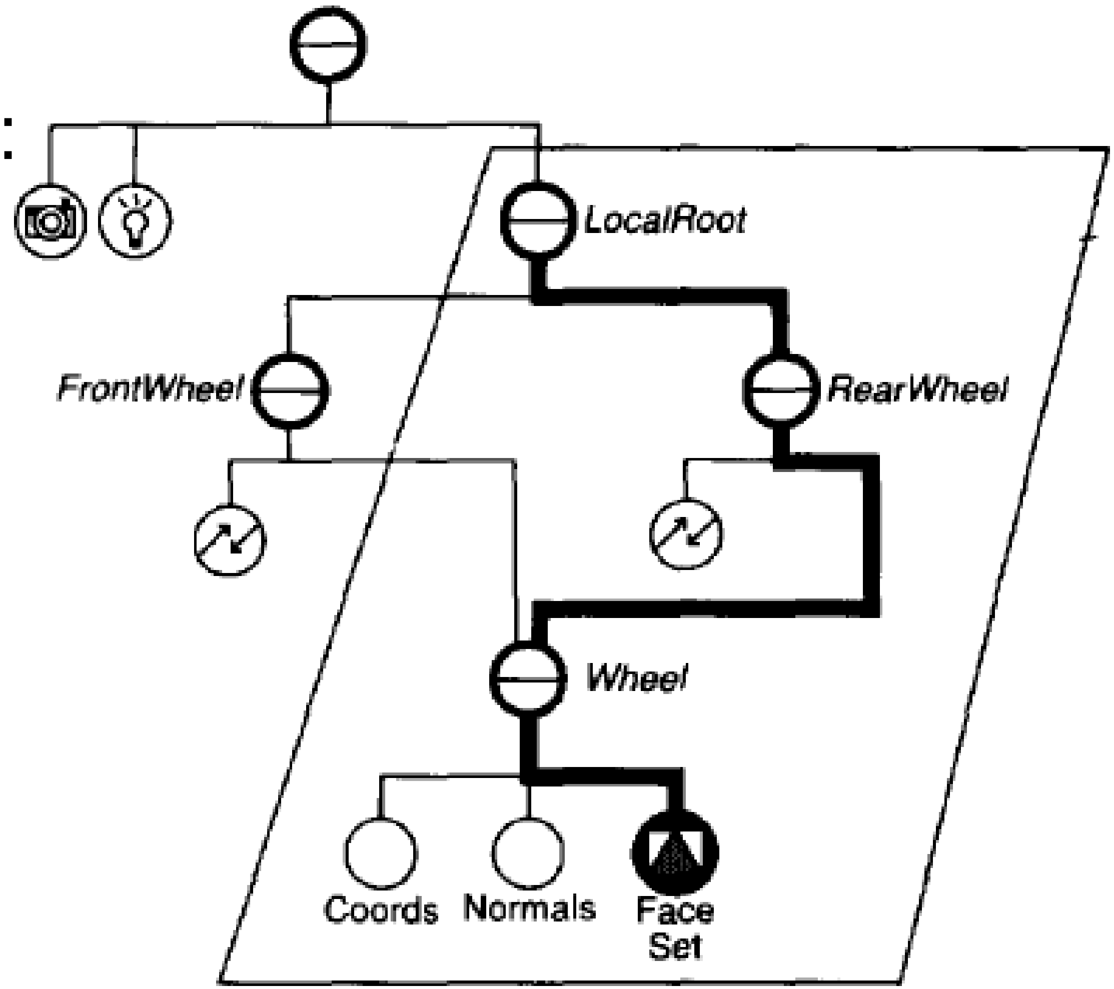
OpenGL Matrix Stack <--> Transformation hierarchy

Hannes
Kaufmann

Instancing (Re-use)

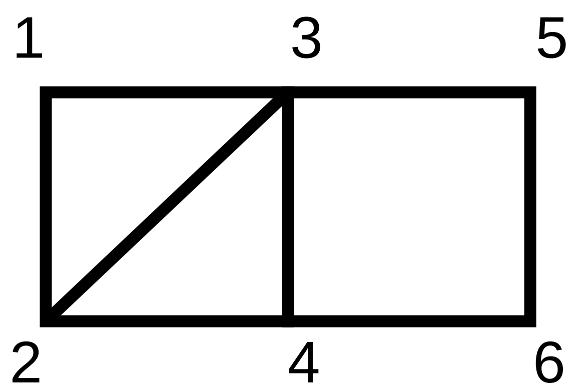
Example: Car

In case of textures:
- Saves texture memory



Hannes
Kaufmann

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}

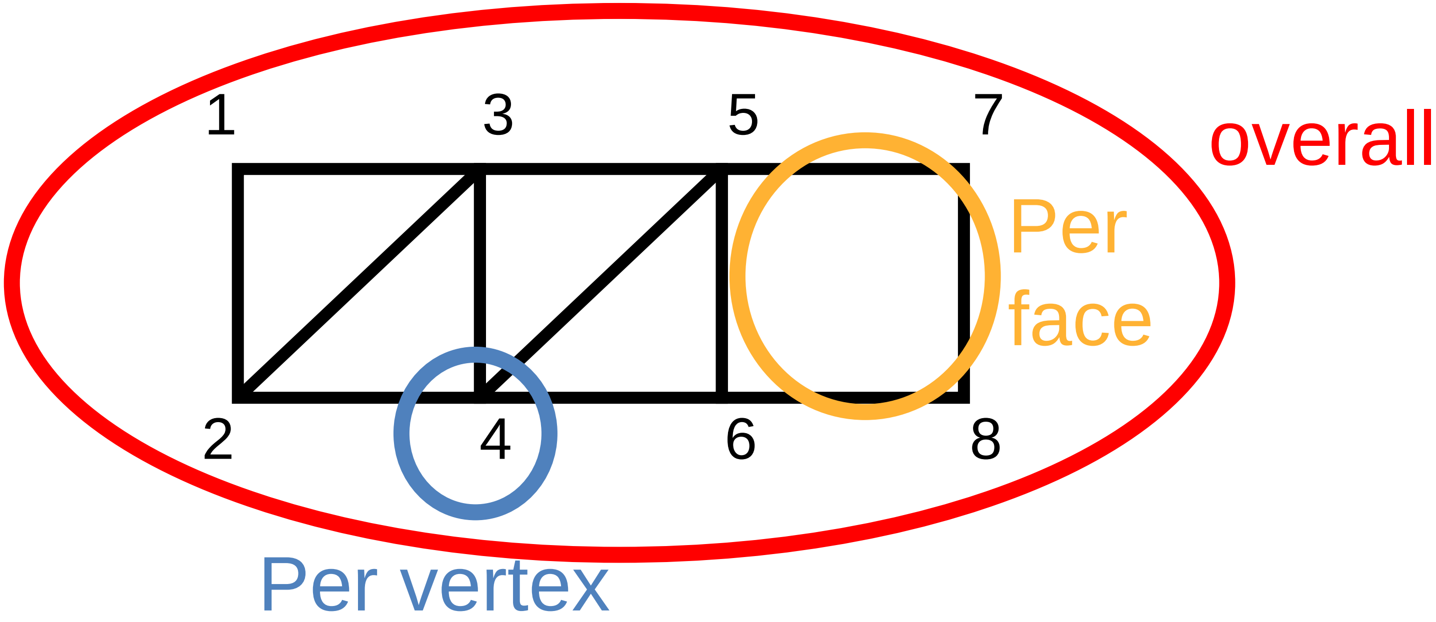
Indexed:

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

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

Hannes Kaufmann

Polygonal Shapes: Attribute

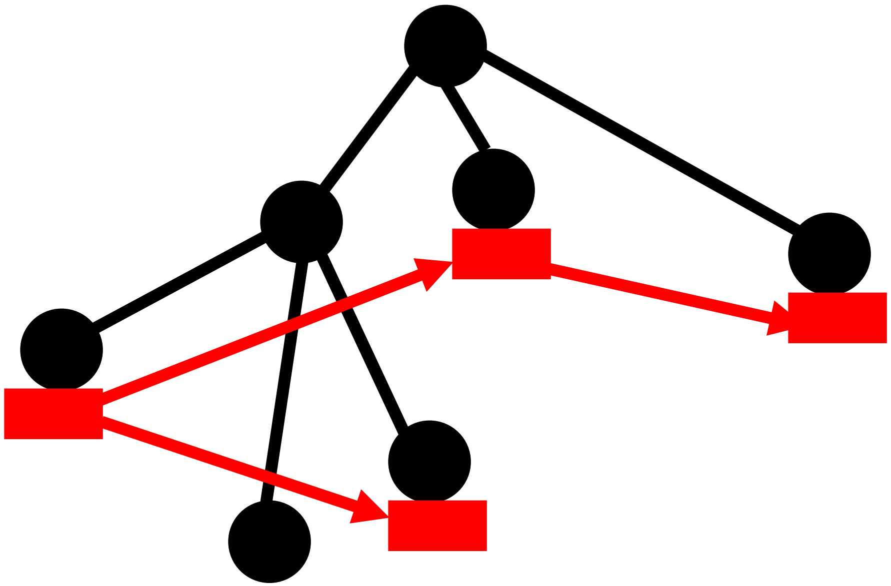


Bindings of attributes

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

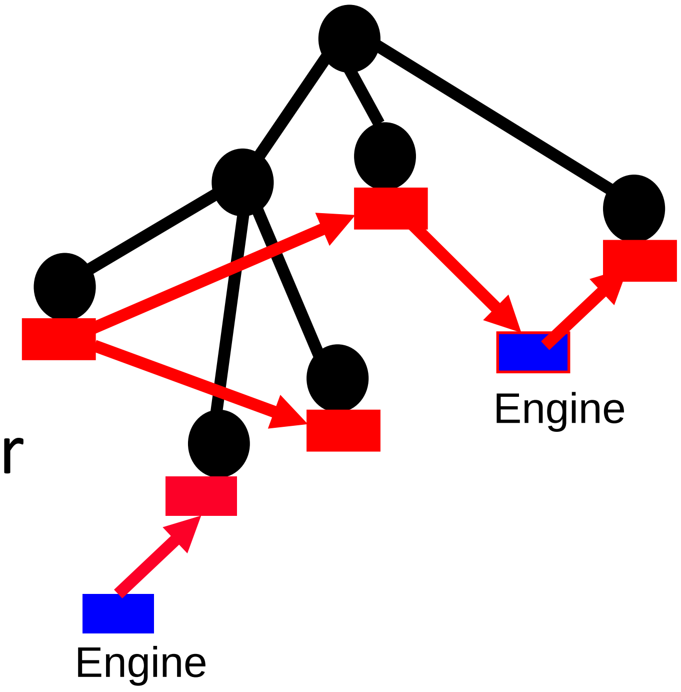
Dependency Graph

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



Engines

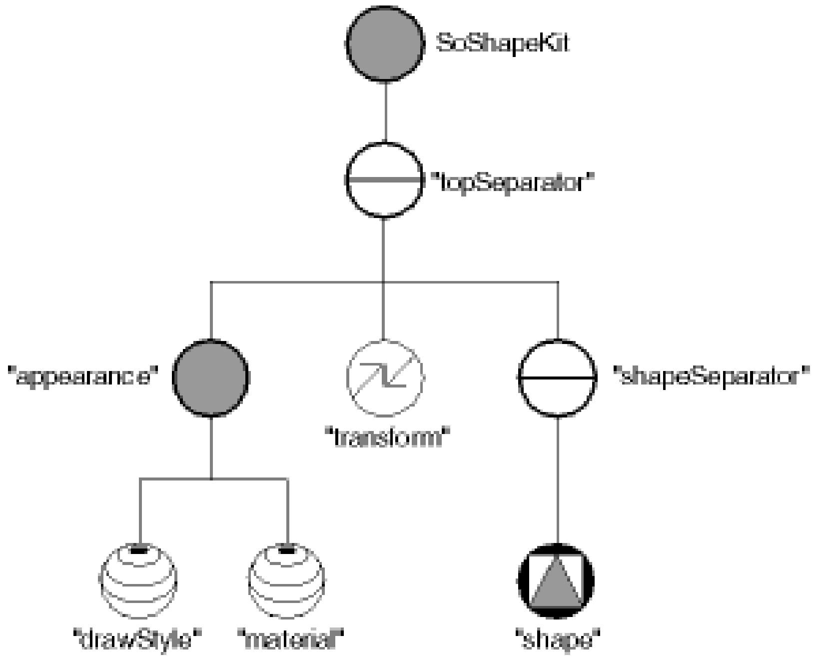
- To model complex dependencies in graph
- TargetField := Engine(SourceField)
- E.g. Calculator, Counter, Type converter, Interpolator, Trigger



Hannes Kaufmann

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

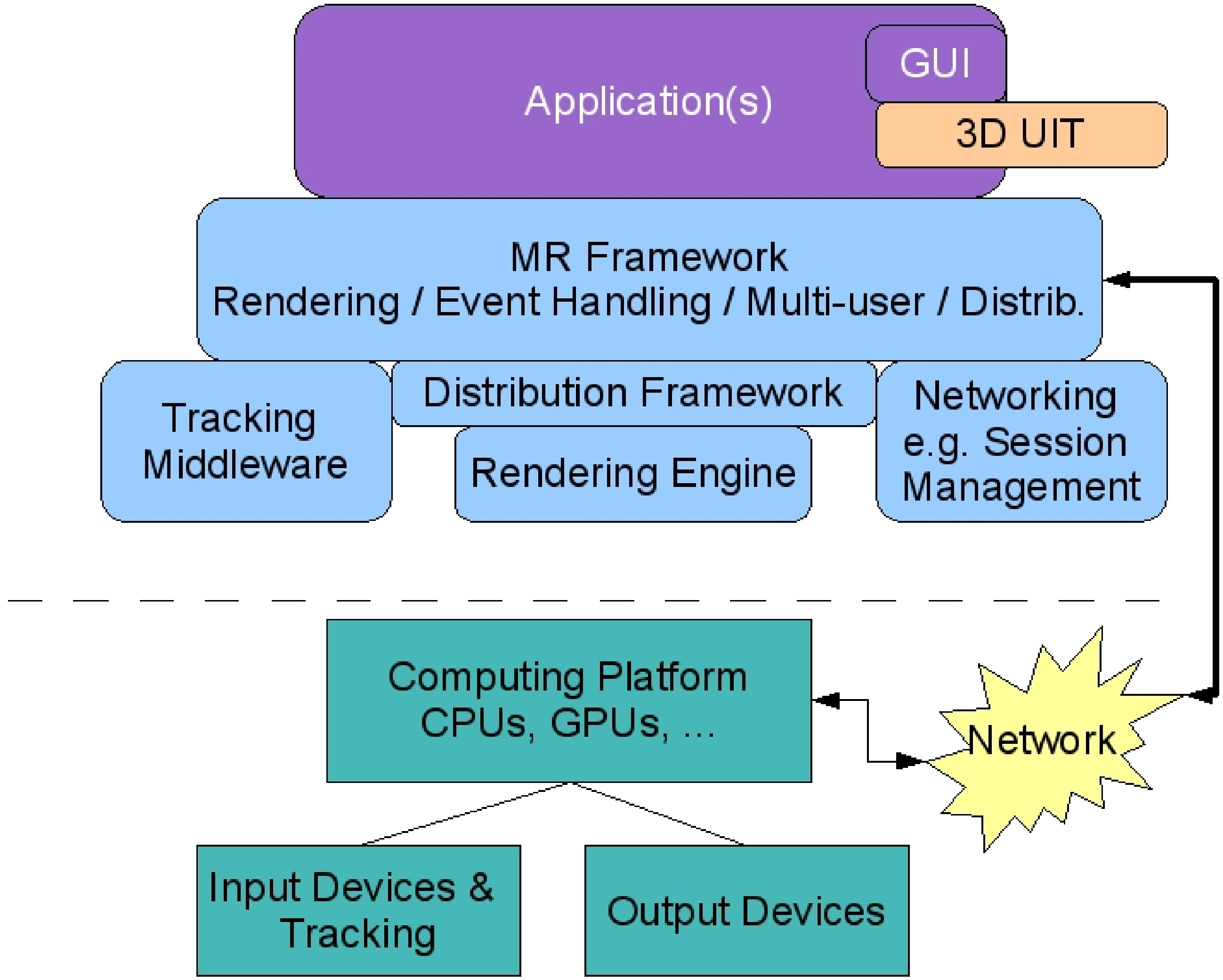
Hannes
Kaufmann

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

Hannes
Kaufmann

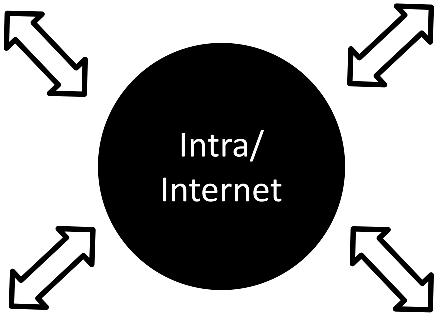
VR/AR/MR System Architecture



Hannes
Kaufmann

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



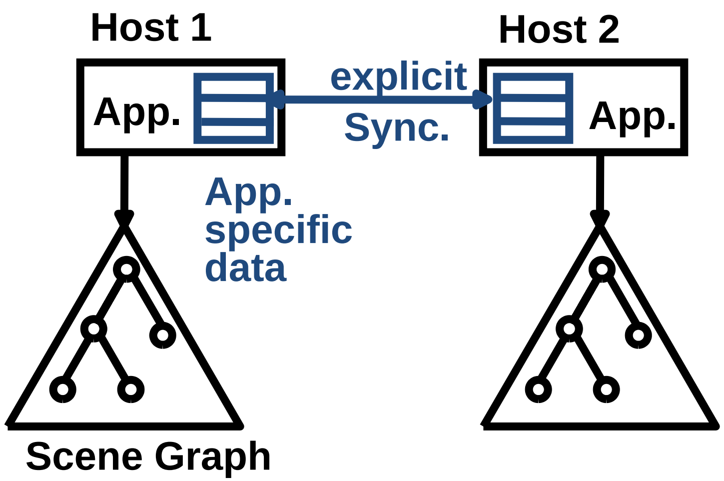
Hannes
Kaufmann

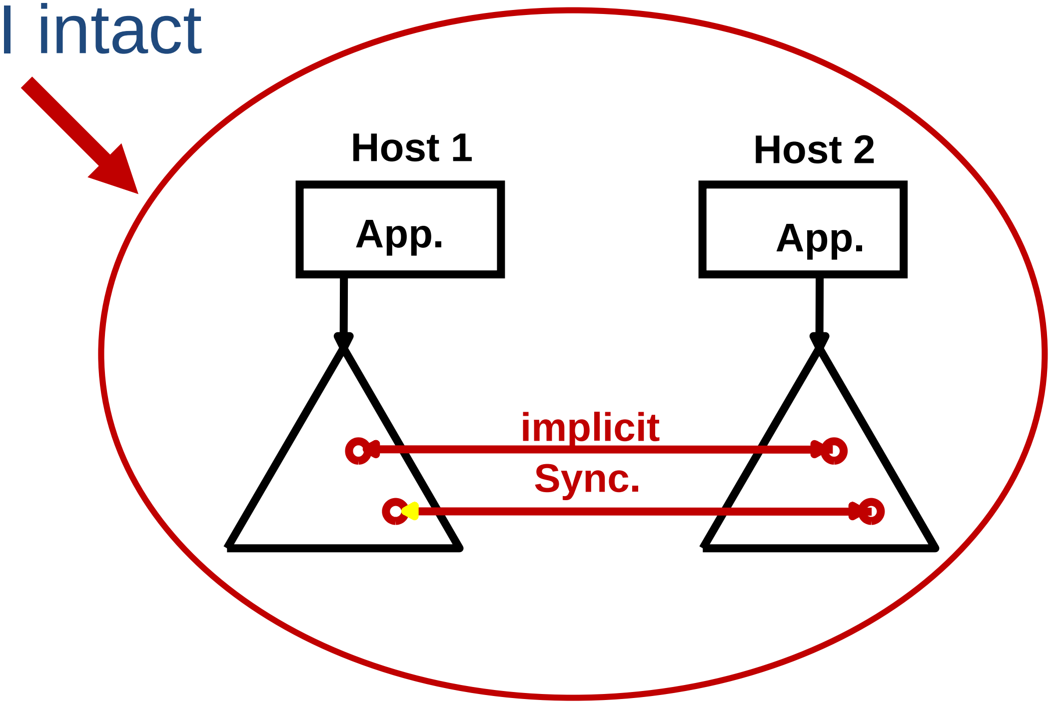
Distributed Shared Scene Graph

- 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

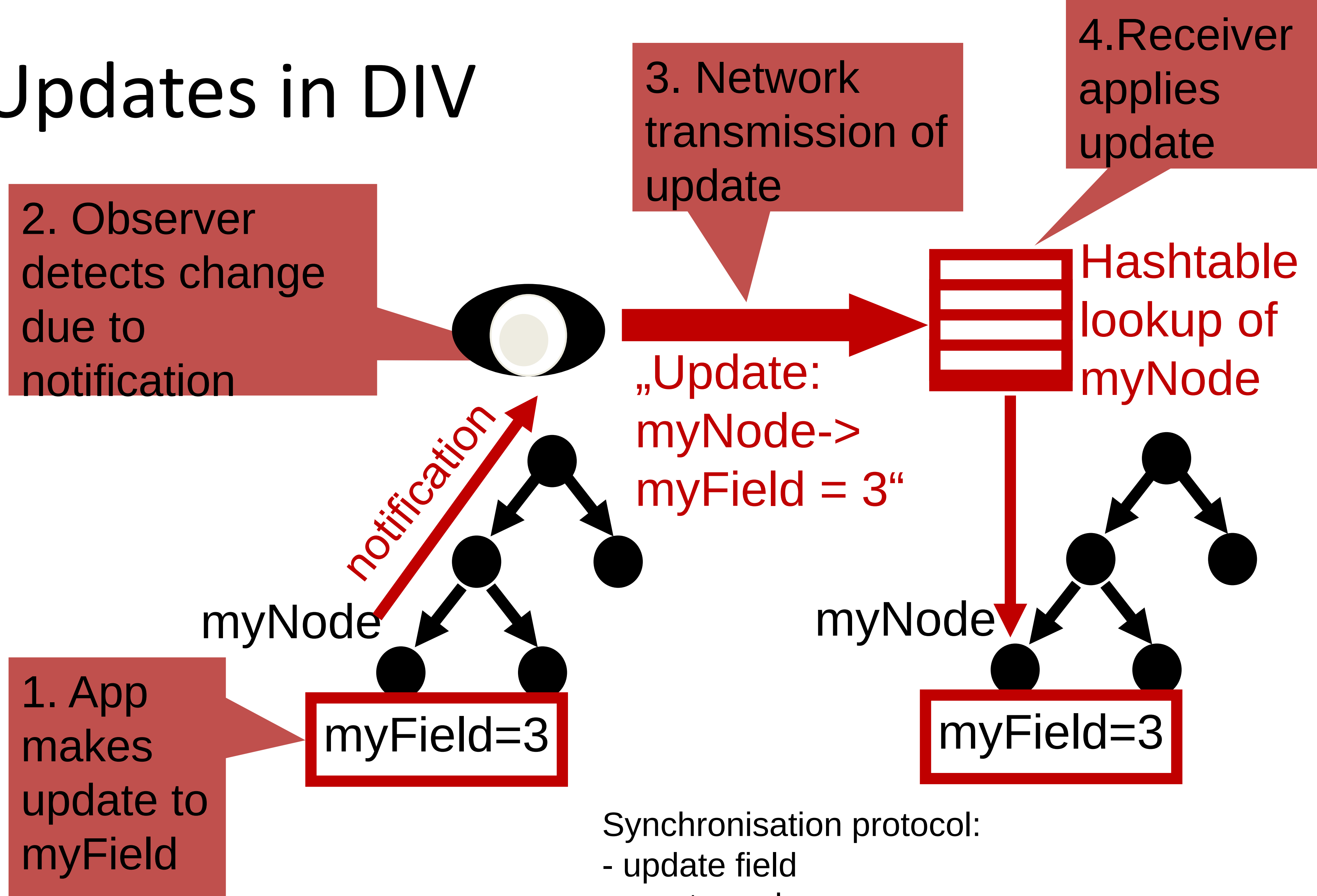




- Dual database (app, scene)
- Optimizations

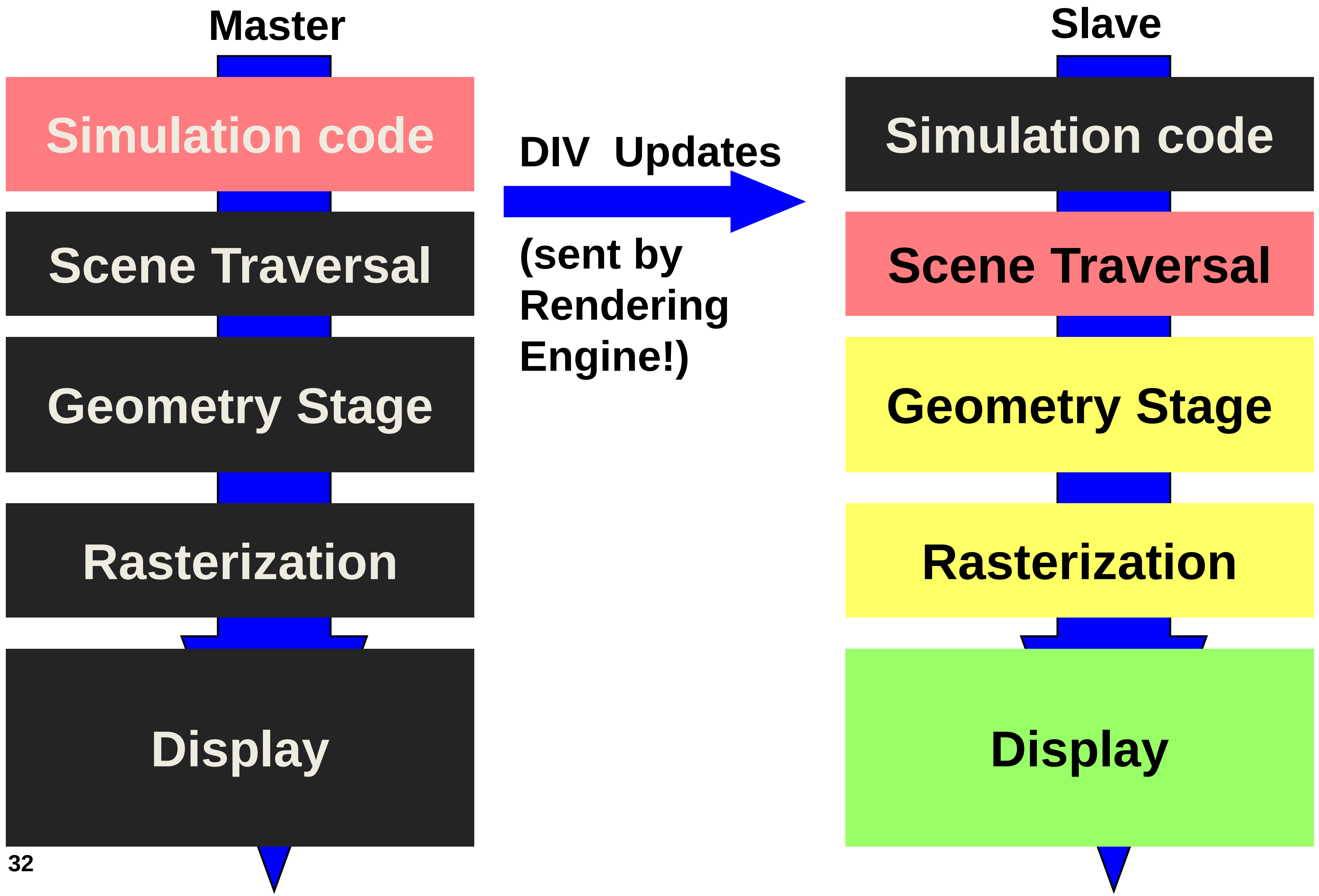
- Distributed shared memory semantics
- Transparent distribution
- E.g.: Avango, Distributed Inventor (**DIV)**

Updates in DIV



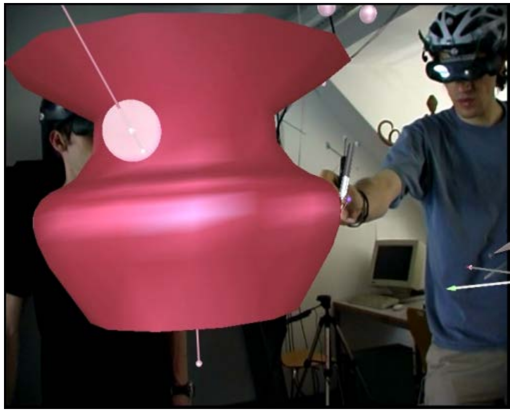
Synchronisation protocol:
- update field
- create node
- delete node
+ some more for for convenience...

DIV - Pipeline



Hannes
Kaufmann

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

Hannes Kaufmann

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

Hannes Kaufmann

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)