Einführung in die Erweiterte Realität

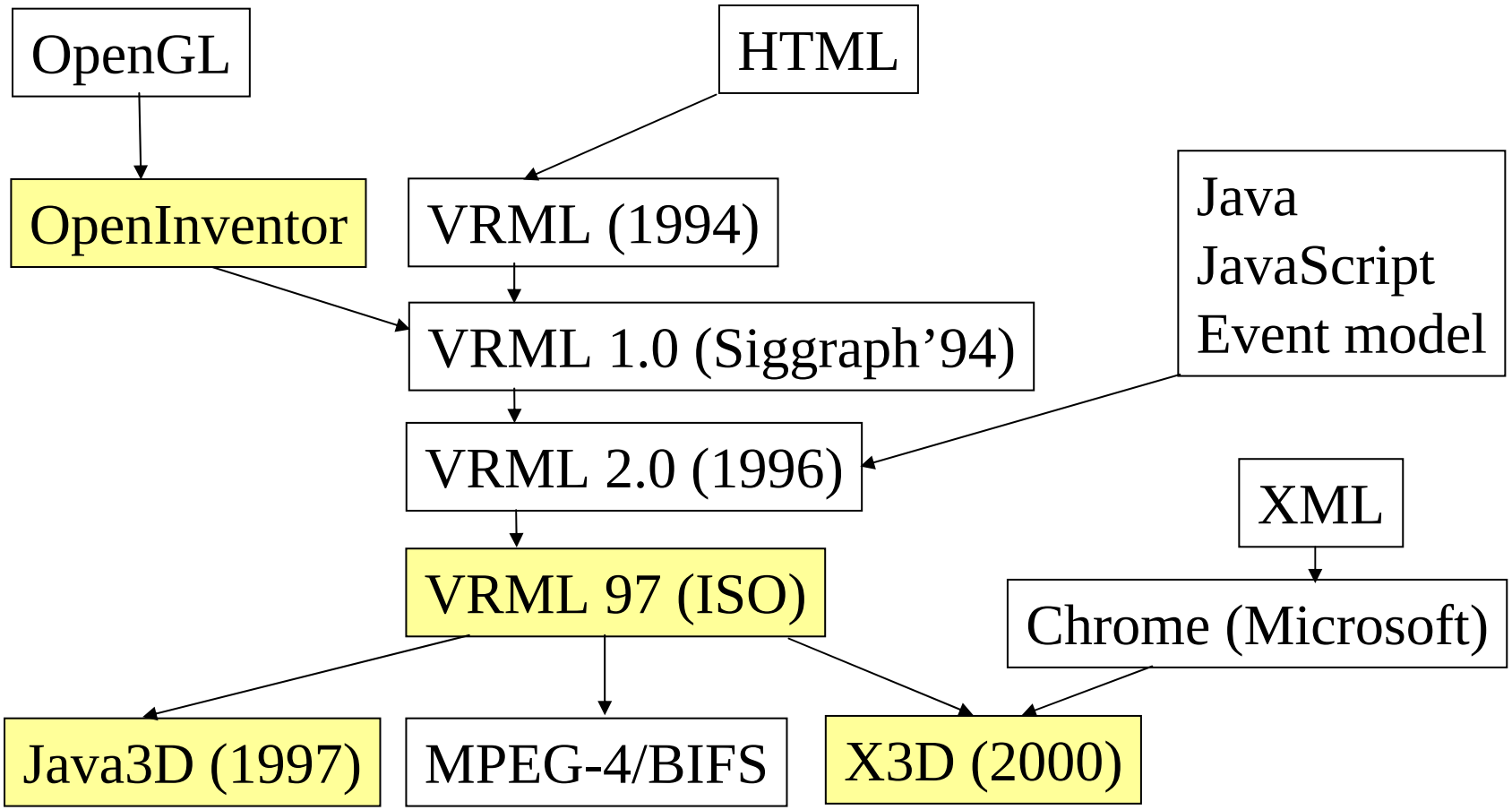
4. APIs

Open Inventor, VRML, X3D, Java3D

Prof. Gudrun Klinker, Ph.D.
Institut für Informatik,Technische Universität München
klinker@in.tum.de

Nov 23, 2004

History of 3D APIs: Web3D



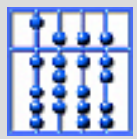
Agenda

1. Open Inventor
2. VRML
3. Java3D
4. X3D

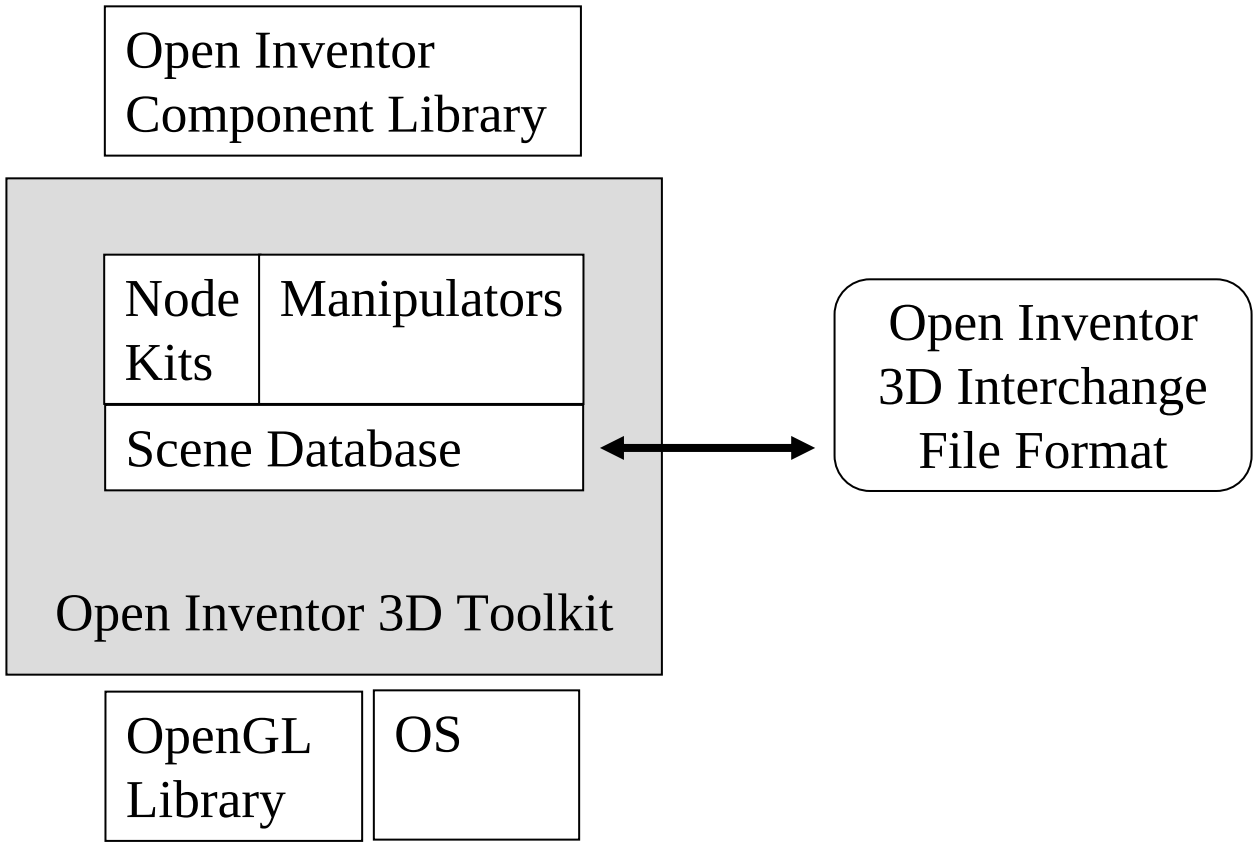
1. Open Inventor Toolkit (OI)

- Object-oriented programming support
  http://techpubs.sgi.com/library/manuals/0000/860-0108-001/pdf/860-0108-001.pdf
  - 3D scene database
  - Set of node kits
  - Set of manipulators
  - Inventor Component Library for Xt

- Coin3D www.coin3d.org
  - Compatible to OpenInventor 2.1, VRML97
  - 3D sound, 3D textures, parallel rendering
  - Multiple platforms: Unix, Linux, Microsoft, Max OS X



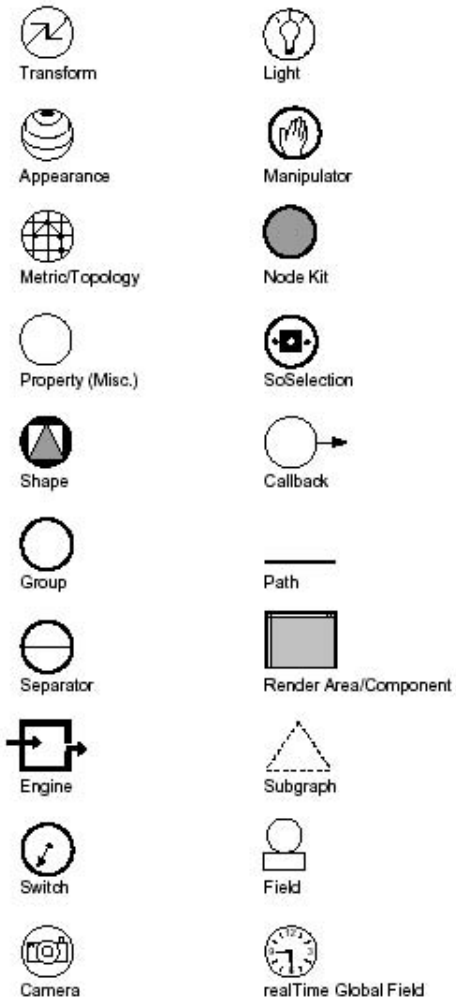
OI Inventor Architecture



OI Node

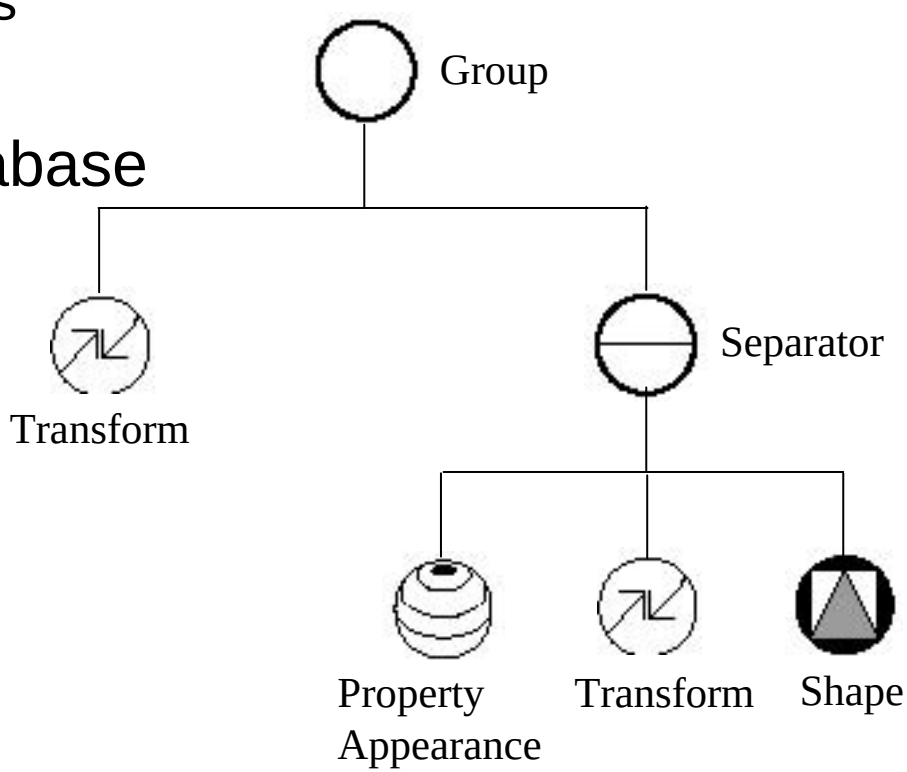
- Basic building block: *node*
- Composed of
  - a number of *fields* of a particular *type*
  - *with associated* actions

Node	Field
SoCoordinate3	point
SoNormal	vector
SoMaterial	ambientColor
	diffuseColor
	specularColor
	emissiveColor
	shininess
	transparency



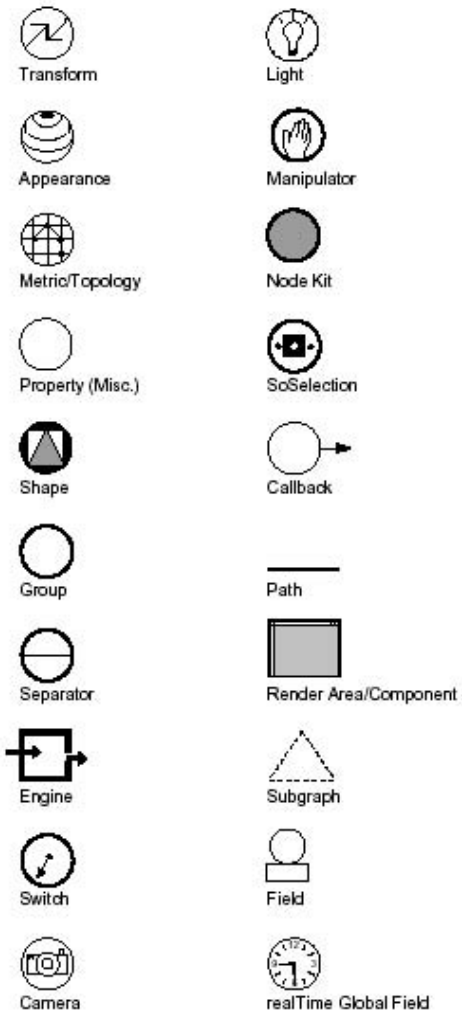
OI Scene Graph

- Scene Graph:
  - Ordered collections of nodes
  - Directed acyclic graph
- Stored in the Inventor database
- Applicable *actions*:
  - Rendering
  - Picking
  - Searching
  - Bounding box comp.
  - Saving
  - …



OI 3D Scene Database

- Node classes:
  - Shapes
  - Properties
  - Groups
  - Cameras
  - Lights
  - Engines
  - Sensors

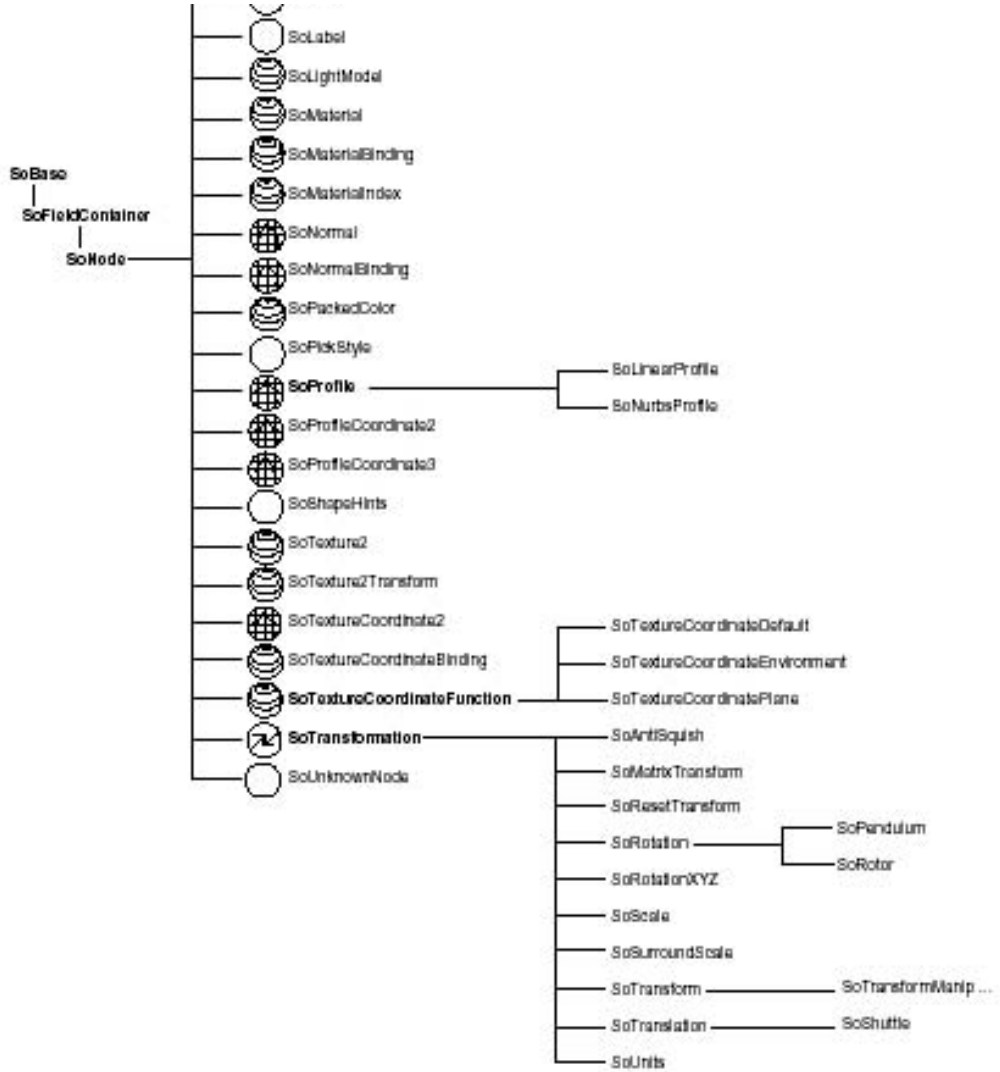


OI Class Tree - Base Classes -

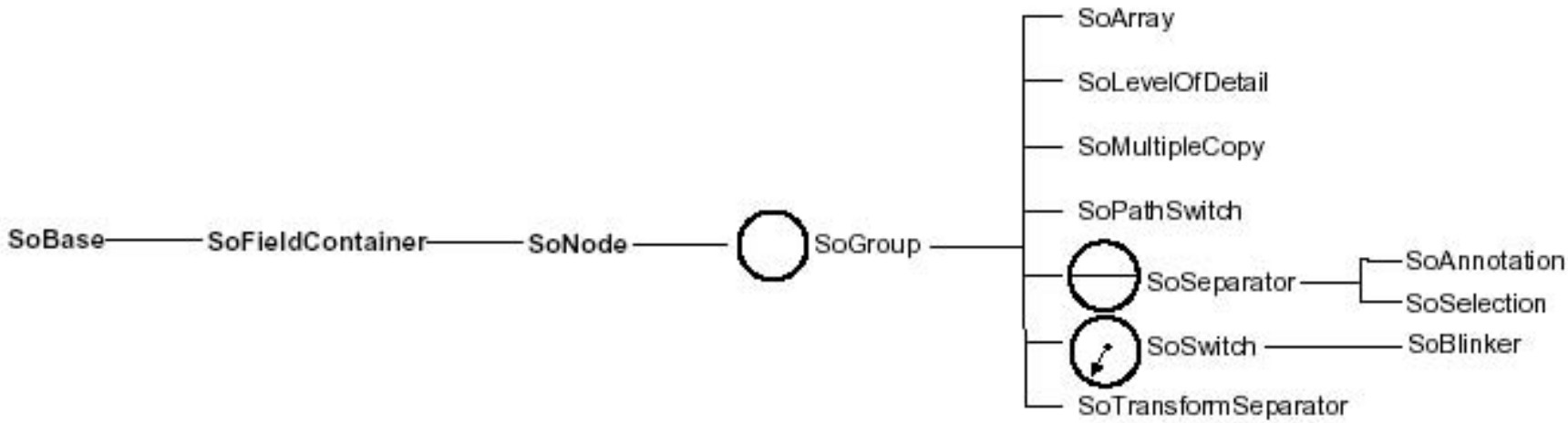
- SoBase
- SoAction
- SoDB
- SoError
- SoInput
- SoOutput
- SoSensor
- SoEvent
- SoDetail
- SoXt
- SoXtComponent
- SoXtClipboard
- SoXtDevice

OI Class tree - example: *shape* -

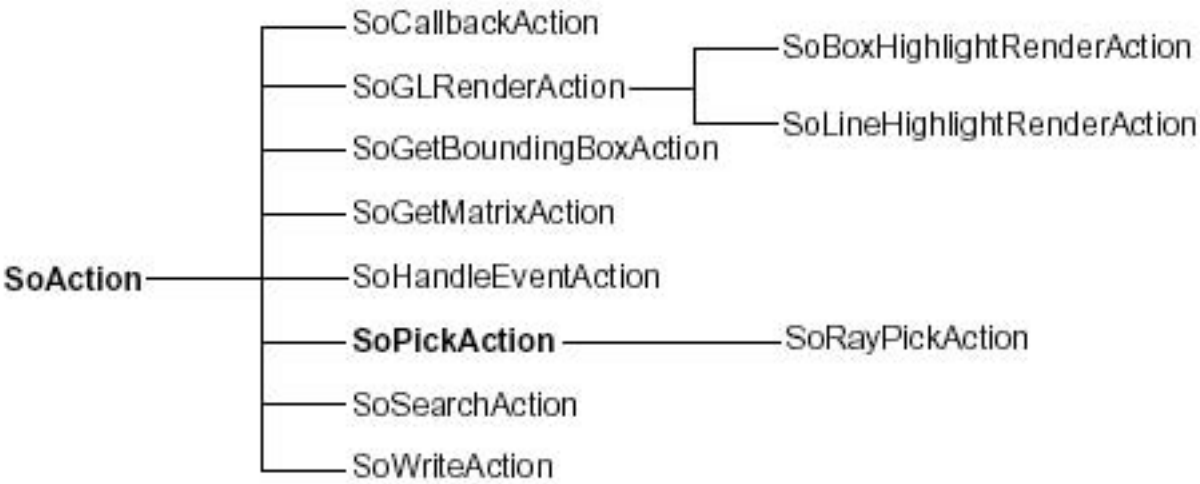
OI Class tree - example: *property* -



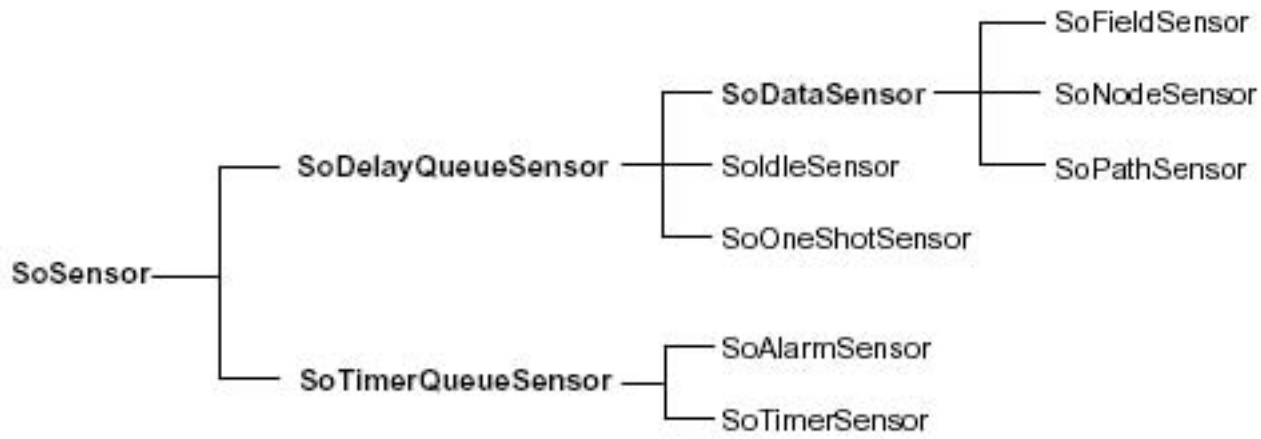
OI Class tree - example: *group* -



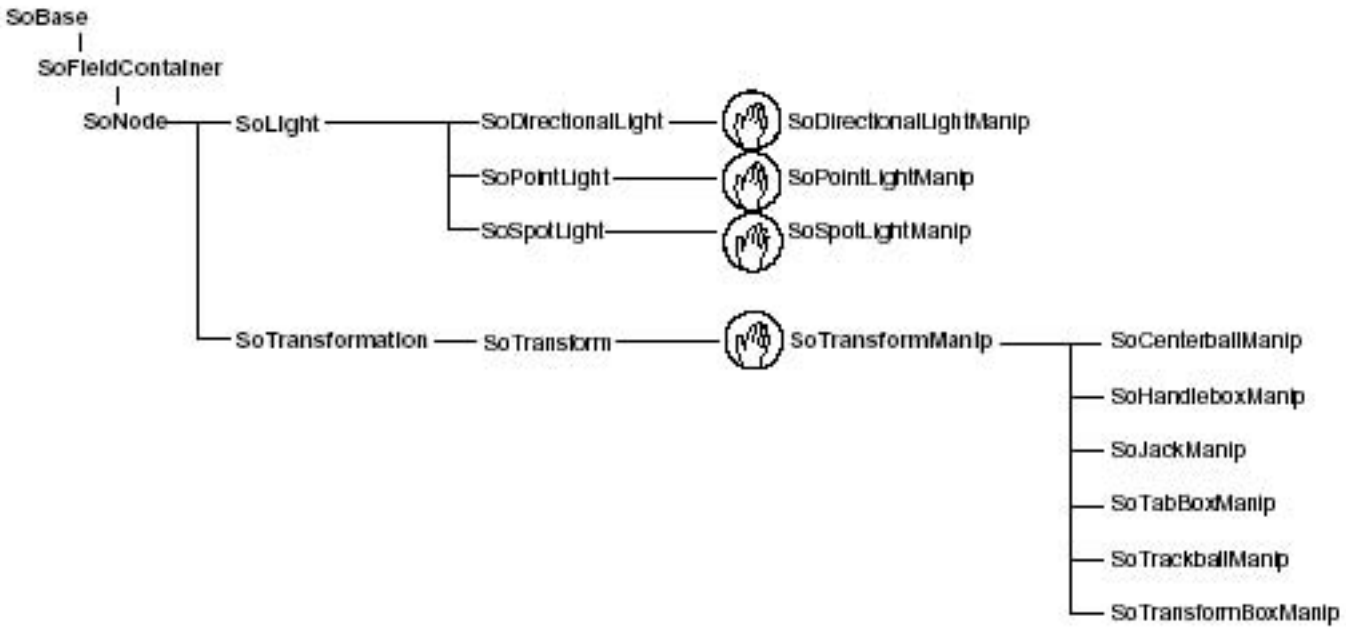
OI Class tree - example: *action* -



OI Class tree - example: *sensor* -



OI Class tree - example: *manipulator* -



OI Manipulators

- Nodes that react to user interface events
- Can be edited directly by the user

- Have a rendering component
- Provide means for translating events into changes in the database

- E.g.: *handle box*

OI Extending the Toolkit

- Callback functions
  - User-written function that is called under certain conditions
  - SoCallback (to make calls to OpenGL)
  - SoCallbackAction (during scene graph traversal)
  - SoEventCallback (for special event handling)
  - SoSelection (to highlight… selected objects)
  - Manipulators
  - SoXt

VRML

Web3D

- Consortium, striving towards
  - making 3D graphics available on the Internet

- Approach: divide graphics into a
  - Model specification (geometry + behavior)
    (file describing a scene graph)
  - Interpreter
    (program rendering the scene according to viewer interactions)

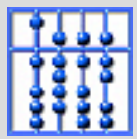
2. VRML

1. Open Inventor
2. VRML and Web3D
    1. General information on VRML
    2. Basic concepts: minimal VRML worlds
    3. Combination of several objects
    4. Model reusability
    5. Levels of detail
    6. Animation
    7. Anchor points
    8. Sensors
    9. Scripts
    10. Prototypes
3. X3D
4. Java3D

2.1. General Information on VRML

- **The VRML repository**: http://www.web3d.org/vrml/vrml.htm
- **Browsers und plug-ins**: e.g. Cosmo Player http://www.cai.com/cosmo/home.htm
- **Book** (excerpts only):
  VRML 2.0 Source Book, Andrea L. Ames, David R. Nadeau and John L. Moreland, 2nd Edition, Wiley & Sons, 1996. ISBN 0-471-16507-7

<Chap. 1>



VRML-Models in the WWW

- http://www.web3d.org/vrml/oblib.htm

- http://www.rdservice.de/german/produkte/VRCreator/library/vrlibrary.htm
- http://home.t-online.de/home/kiwano6/models.htm
- http://www.rccad.com/Gallery.htm

- for recent demos and examples, see
  http://wwwbruegge.in.tum.de/people/klinker/MMT/Einfuehrungen/VRML/
  http://wwwbruegge.in.tum.de/teaching/ss00/ar/vrml/

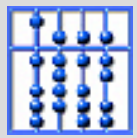
2.2. Basic Concepts: Minimal VRML worlds

VRML Files

ASCII Files: <my_world>.wrl

- **VRML header**: #VRML V2.0 utf8
- Comments: #
- **Nodes**: Shapes, transformations, timers, interpolators, sensors, scripts, ...
- **Routes**

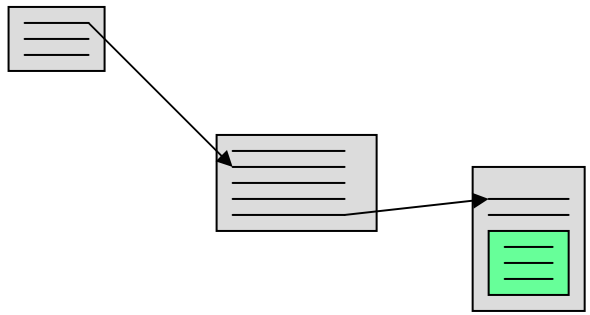
<Chap. 2>



Object Descriptions in VRML

Nodes (nested, grouped) with

- **Fields** and **Field Values** of specified **Field Value Types**
  - **Inputs** and **Outputs**
  - **Bounding Boxes**



Examples:

- Shape nodes

```
Shape {
    appearance SFNode,
    geometry SFNode
}
```

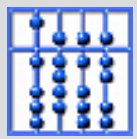
- Other nodes: transformations, timers, interpolators, sensors, scripts, …

<Chap. 2, 3>

Geometry Nodes - Cylinder -

```
#VRML V2.0 utf8

# default cylinder
Shape {
    geometry Cylinder {
    } # Cylinder
} # Shape
```

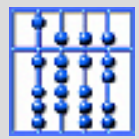
```
#VRML V2.0 utf8

# cylinder of specified size
Shape {
    geometry Cylinder {
        height 4.0
        radius 2.0
    } # Cylinder
} # Shape
```

Geometry Nodes - Box, Sphere -

```
#VRML V2.0 utf8

# box of specified size
Shape {
    geometry Box {
        size 2.0 1.0 4.0 # dx,dy,dz
    } # Box
} # Shape
```

```
#VRML V2.0 utf8

# sphere of specified radius
Shape {
    geometry Sphere {
        radius 2.5
    } # Sphere
} # Shape
```

Geometry Nodes - Cone, Text -

```
#VRML V2.0 utf8

# cone of specified size
Shape {
    geometry Cone {
        bottomRadius 5.0
        height 1.5
    } # Cone
} # Shape
```

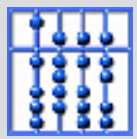
```
#VRML V2.0 utf8

# text of specified font
Shape {
    geometry Text {
        string ["Hello World","Cheers"]
        fontStyle  FontStyle {
            family "SERIF"
            style "BOLD"
            size 0.5
             justify "MIDDLE"
        }
    } # Text
} # Shape
```

Geometry Nodes - Point Sets, Indexed Line Sets -

```
#VRML V2.0 utf8

# set of 3D points
Shape {
    geometry PointSet {
        coord Coordinate {
            point [
                 0.0   0.0   0.0,  # index 0
                 0.0   1.0   0.0,   # index 1
                 1.0   1.0   0.0,   # index 2
                 1.0   0.0   0.0    # index 3
            ] # point
        } # Coordinate
    } # PointSet
} # Shape
```

```
#VRML V2.0 utf8

# set of line indices
Shape {
    geometry IndexedLineSet {
        coord Coordinate {….}
        coordIndex [
            0, 1, 2, -1,
            0, 2, 3
        ] # coordIndex
    } # IndexedLineSet
} # Shape
```

```
#VRML V2.0 utf8

# set of line indices
Shape {
    geometry IndexedFaceSet {
        coord Coordinate {….}
        coordIndex [
          0, 1, 2, -1,
          0, 2, 3
        ] # coordIndex
    } # IndexedLineSet
} # Shape
```

Appearance Nodes - Materials/Colors -

```
#VRML V2.0 utf8
Shape {
        geometry SFNode,
        appearance Appearance {
            material Material {
                ambient Intensity 0.2
                 diffuse Color       0.8   0.8   0.8
                  specular Color     0.0   0.0   0.0
                  ....
            } # Material
            texture SFNode
            textureTransform SFNode
        } # Appearance
} # Shape
```

2.3. Combination of Several Objects

- Inlining of other files

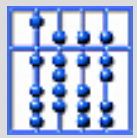
  Inline {url [..., ...], ...}

- Geometric Transformations

  Transform {
  children [...]
  translation ... ...
  rotation ... ...
  scale ... ...
  }

<Chap. 12>

<Chap. 5-7>

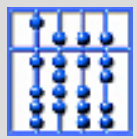


Geometric Transformations - Box, Sphere -

```
#VRML V2.0 utf8

# cross in xz-plane; translated in x-direction, rotated 45 deg around y-axis
Transform {
    children [
        Shape { geometry Box { size 1.0 1.0 4.0 } } # Box1
        Shape { geometry Box { size 4.0 1.0 1.0 } } # Box 2
    ]
    translation   10.0   0.0   0.0
    rotation       0.0   0.0   1.0    0.785   #  in radians; pi/4= 45 degrees
    scale          1.0   0.1   1.0
} #Transform
```

Example: Sphere plus Cylinder

```
#VRML V2.0 utf8

# 1. Object
Inline {url "sphere.wrl"}

# 2. Object
Transform {
    translation 0.0 2.0 0.0
    rotation 0.0 0.0 1.0 -0.524    # radians; 30 degrees
    children [ Inline {url "cylinder-c.wrl”} ]
} # Transform
```

Example: Blocks World (Baufix)

- Used as library for further demos

```
#VRML V2.0 utf8
Transform {
    translation -6.0 0.5 0.0
    children [ Inline {url "yellow-box1.wrl"} ]
} # Transform
Transform {
    translation -4.5 1.0 0.0
    children [ Inline {url "blue-box2.wrl”} ]
} # Transform
...
```

2.4. Model Reusability

Reuse of Objects

- Inlining
- DEF and USE

```
DEF Green-Box3
    Inline {url "../blocks-
       world/green-box3.wrl”
    } # Inline cylinder
...
USE Green-Box3
```

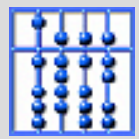
<Chap. 2>

Multiple Use of Groups of Objects

- Groups

```
Group {
    children [
        ...
    ]
    ...
}
```

<Chap. 3, 11>

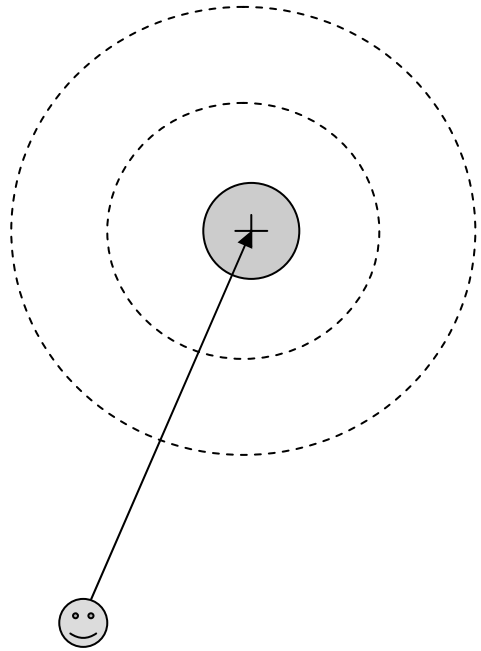


Example: Replicated Model

```
DEF Rep-Structure Group {
    children [
        DEF Baufix-Structure Inline {url "structure-group.wrl”} # 1. structure
        Transform {                                              # replicated structure
            translation 0.0 0.0 -2.0
            children [USE Baufix-Structure]
        } # Transform 2. Structure
        # ... Further structures ...
    ] # children
} # group
Transform {
    translation 5.0 0.0 0.0
    children [USE Rep-Structure]                                 # replicated group
}
```

2.5. Level of Detail, Switch

- Geometric objects with varying amounts of complexity
  - selective presentation depending on the distance between the viewer and the object center
  - LOD {
    center 0.0 0.0 0.0
    range [7.5 15.0]
    level [
    Inline {url “obj-highdetail.wrl”}
    Inline {url “obj-mediumdetail.wrl”}
    Inline {url “obj-lowdetail.wrl”}
    ]
    }

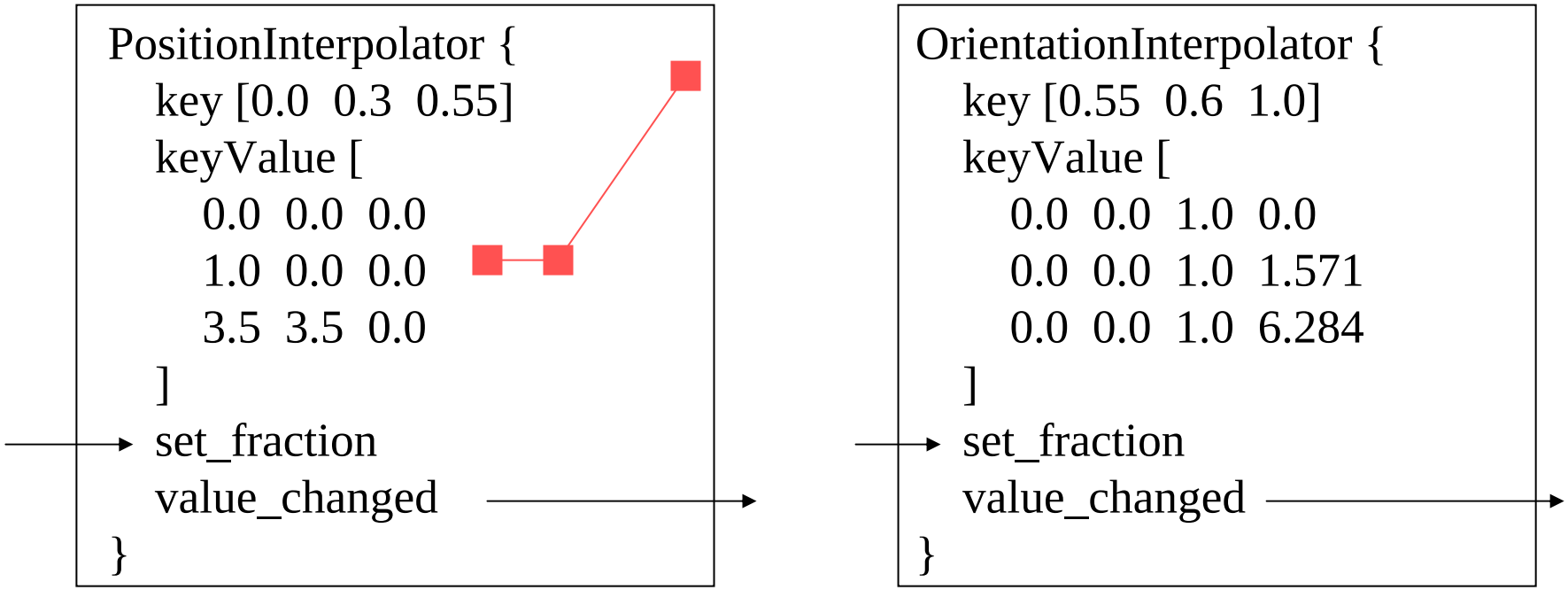


- Switch to select between several shapes
  - Switch {
    whichChoice -1
    choice []
    }

<Kap. 25>

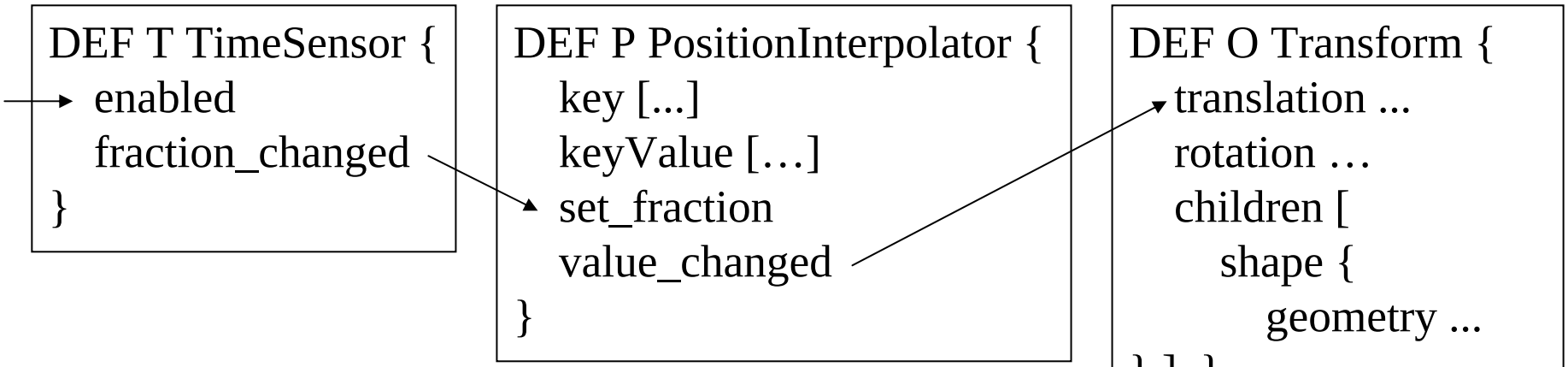
2.6. Keyframe Animation

- VRML: linear Interpolation between n “key frames”



ROUTEs

- Connections between nodes in order to forward events



ROUTE T.fraction_changed TO P.set_fraction

ROUTE P.value_changed TO O.set_translation

2.7. Anchors

- Transition from one world into another (and back) via Anchors.
- (e.g.: Dungeon Worlds, different levels)

```
Anchor {
    url [“…”, “...”, ...]
    children […]
    description “…”
    ...
}
```

<Kap. 28>

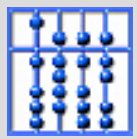
Anchors for AR

- “hot-spots”
  - as a 3D menu (e.g.: next to a real object) supplying dynamic, stepwise construction information
  - contained within the virtual models, providing additional information for specific object parts
  - contained within the virtual models, providing alternate object presentations
  - embedded as virtual camera symbols within the scene, representing alternate (video-based) views of the scene (--> tele-presence)

2.8. Sensors

- Sensors for various interaction modes:
  - cursor position (TOUCH): isOver, isActive
    DEF Touch-Green TouchSensor{}
    ROUTE Touch-Green.touchTime TO Clock.set_startTime
  - cursor motion (MOTION)
    - CylinderSensor
    - PlaneSensor
    - SphereSensor
      DEF Spin-Cylinder SphereSensor{}
      ROUTE Spin-Cylinder.rotation_changed TO Red-Cylinder2-b.set_rotation
  - when nested sensors are used, the innermost is selected
  - several modi for progress (Offsets)

<Kap. 9>

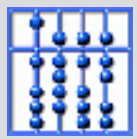


More on Sensors

- Viewer proximity (for cursor motion)
  - visibility
    - VisitilitySensor{}
  - viewer proximity
    - ProximitySensor{}
  - viewer collision
    - Collision node

```
DEF MyCollision Collision {
    children []
    collide TRUE
}
ROUTE MyCollision.collideTime TO OuchSound.set_startTime
...
```

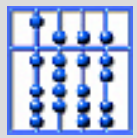
<Kap. 27>



2.9. Skripts

- Connection VRML -> Java for generating more complex behavior
  - complex motion paths of objects (e.g., modelling a realistic walking motion)
  - intelligent reactions of objects
  - communication in distributed systems (e.g., for multi-player games)
- EAI: External authoring interface

<Kap. 30>



Skripts

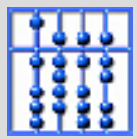
- Example: Motion Interpolator

VRML:

```
DEF Mover Script {
    url "move1.js"
    eventIn    SFFloat   set_fraction
    eventOut  SFVec3f  value_changed
}
...
ROUTE Clock.fraction_changed TO Mover.set_fraction
ROUTE Mover.value_changed TO  Red-Cylinder2-b.set_translation
```

Java file (move1.js):

```
// move a shape in a sinusoidal path
function set_fraction (fraction,eventTime) {
    value_changed[0] = 6.0 + Math.sin (fraction * 6.28); // x component
    value_changed[1] = -2.0; // y component
    value_changed[2] = -2 +fraction*4; // z component
}
```

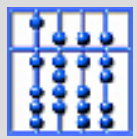



2.10 Prototypes

- Scheme to extend VRML's core capabilities
- Creation of libraries of reusable custom nodes in external files

```
PROTO Saturn [
    field SFColor planetColor   1 0 0
    field SFColor ringColor      1 1 0
]   {
    … regular VRML code…
}

Saturn {
    planetColor   0 1 0
    ringColor      0 0 1
}
```

```
EXTERNPROTO Saturn [
    field SFColor planetColor
    field SFColor ringColor
]   "saturn.wrl"



Saturn {
    planetColor   1 1 0
    ringColor      0 1 0
}
```

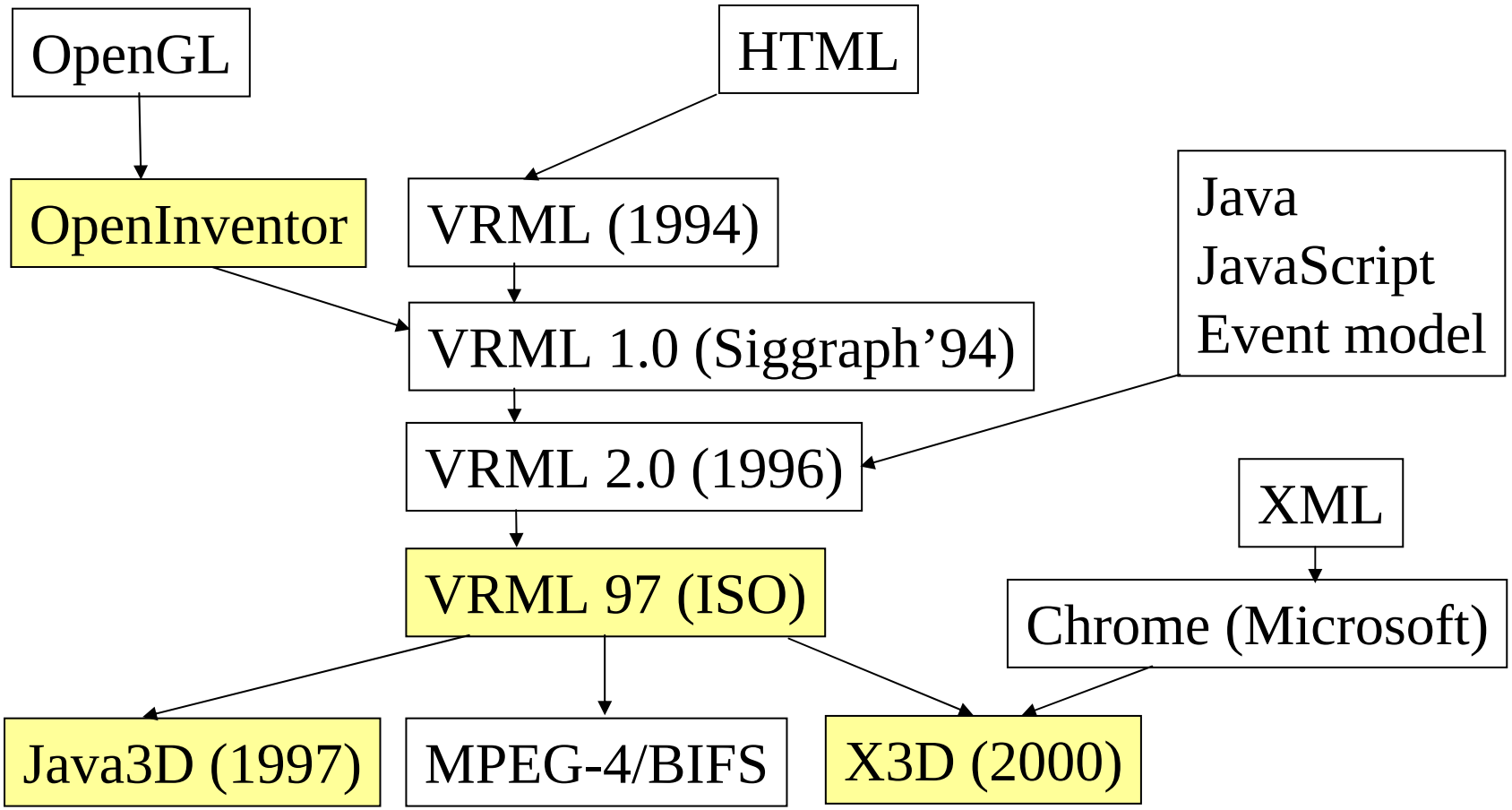



Shortcomings

Web3D: Shortcomings of the early days

- Early limitations:
  - Limited bandwidth
  - Limited platforms
  - Limited content authoring tools
- Gordon Moore's law:
  - Computer capacity of mircrochips will double every 1.5 years without increasing production costs
- Geoffrey Moore's laws:
  (when creating novel technology)
  - Create a *tornado* effect: identify a "*beachhead*" application
  - The force that powers emerging market tornadoes is *discontinuous innovation* (discontinuity)
  - Don't introduce a discontinuity inside another discontinuity

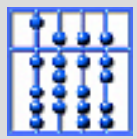
History of 3D APIs: Web3D



Java3D

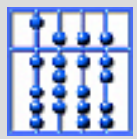
Agenda

1. Open Inventor
2. VRML
3. Java3D
4. X3D

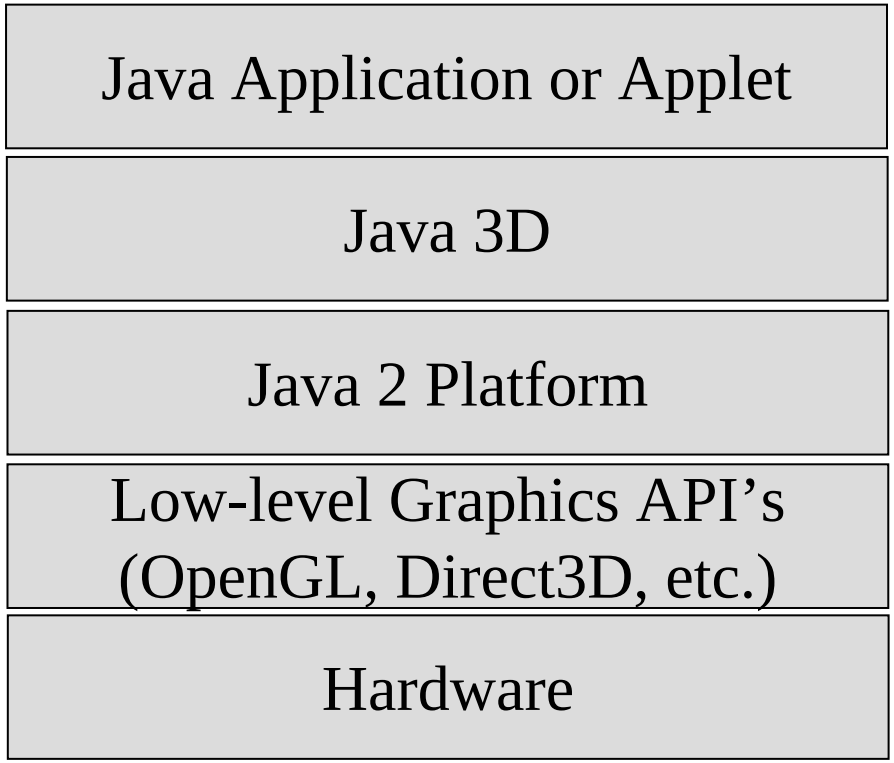


3. Java3D

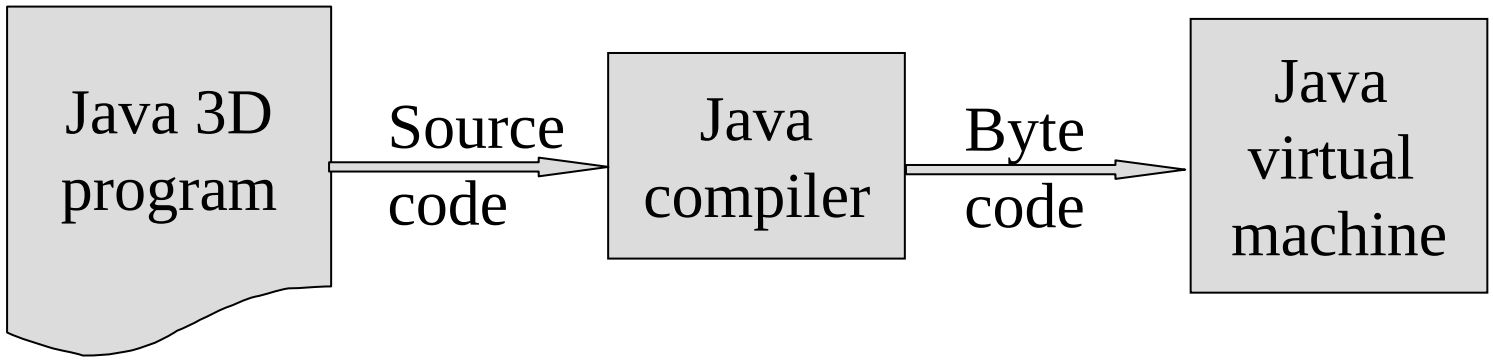
- Collection of Java classes
- High-level API for interactive 3D development
- Design goals:
  - Integration with Java
  - High performance
    - Layered on efficient low-level APIs (OpenGL, Direct3D)
    - Support for hardware acceleration
    - Scene graph structure with potential for optimizations
    - Multiprocessor rendering
  - Support for runtime “loaders”
  - Support for a rich suite of CRITICAL capabilities



Java layers

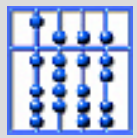

Java Application or Applet
Java 3D
Java 2 Platform
Low-level Graphics API's
(OpenGL, Direct3D, etc.)
Hardware

Java 3D System



Java Media API Family

- Java 2D
- Java 3D
- Java Advanced Imaging (JAI)
- Java Media Framework (JMF)
- Java Shared Data Toolkit (JSDT)
- Java Sound
- Java Speech
- Java Telephony



Java 3D Packages

- All programs based on classes found in packages
  - javax.media.j3d (“core Java 3D classes”)
  - javax.vecmath
  - com.sun.j3d
    - com.sun.j3d.util
      (content loaders, scene graph assembly, geometry, convenience classes)

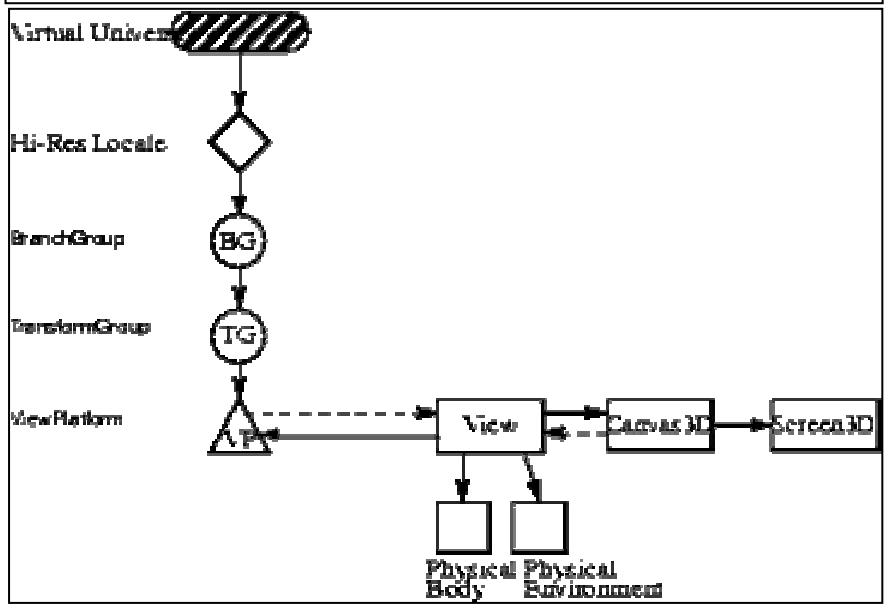
Key Java 3D Features

- Scene graph programming model
  - Scene graph superstructure with
    - One VirtualUniverse node (“root”)
    - One or more Locales
  - Regular scene graphs
- Rendering control
  - Immediate mode
  - Retained mode
  - Compiled-retained mode
- Scalable performance
  - Scene graph optimizations (view culling, occlusion culling, execution culling, behavior pruning, parallel graph traversal, parallel rendering)
  - Multithreading
- Behavior nodes
  - Execution culling (3D behavior scheduler)

Key Java 3D Features (cont.)

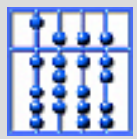
- Generic input devices
  - Polled input
  - Continuous input
  - Input-device scheduler
- Geometry compression
  - Byte code
- Convenience and utility classes
- Versatile view model
  - Canvas3D
  - Screen3D
  - View
  - PhysicalBody
  - PhysicalEnvironment
- Camera-based view model
  - Head-tracking
  - Stereo, Monitor, HMD, Cave

Distinction between
- virtual world (“view platform”)
- real world (“view”)



Web3d

X3d

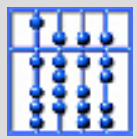


Agenda

1. Open Inventor
2. VRML
3. Java3D
4. X3D

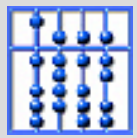
4. X3D

- Key problems with VRML97
  - VRML browsers too large for internet browsers
  - Downsizing of graphics industry
  - In the right place at the wrong time: PCs too slow
  - Source code too big and ugly
  - Conformance concerns
  - VRML objects are not fully integrated into web pages
  - No enforcement of progressive rendering/streaming techniques
  - No proper synchronization with sound
  - GeoVRML needs high precision and a technique to deliberately load and discard data



X3D Requirements

- Primary target applications
  - E-commerce product and technology demonstration
  - Visual simulation
  - Database visualization
  - Advertising and web page animation
  - Augmented news and documentaries
  - Training
  - MultiUser
- Market requirements
  - Interoperation with other standards (XML, XHTML, MPEG-4)
  - Tighter media integration
  - Improved visual quality
  - Component-based approach
  - VRML97 compatibility
  - File formats (streaming, binary file format, text-based format)
  - Time to market

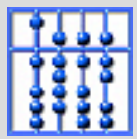


X3d Requirements

- Technical requirements
  - General
    - Components/Levels/Profiles
    - Integrate with existing Web and media standards
    - RealPlayer
    - Platform independence
  - Core
    - Unified typing system
    - Namespaces
    - High-precision numerical representation
    - Deterministic design
    - Powerful capabilities ->

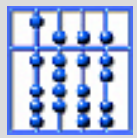
X3D Requirements

- Technical requirements (cont.)
  - Core (cont.)
    - Powerful capabilities
      - Extensive scene graph traversal/manipulation
      - Full access/discovery of dynamic objects
      - Advanced picking and intersection
      - Collision detection/behavior
      - Navigation control
      - Input-device feedback
      - Rendering methods
      - Persistence of state
      - Embedded runtime environment interfaces
      - Runtime context to determine platform/browser/user-specific features
      - Notification for completion of asynchronous operations
      - Debugging support



X3D Requirements

- Technical requirements (cont.)
  - Core (cont.)
    - Consistent and understandable (specify X3D with IDL)
    - Extensible and object oriented (polymorphism, inheritance)
    - Write-once/Run-anywhere
    - Interoperate with standard API models
    - Synchronization
    - Fast content delivery
    - Authoring/publishing file format encodings
    - Author control over tradeoffs
    - Metadata
    - Scene graph optimization hints



X3D Requirements

- Technical requirements (cont.)
  - End-user experience (runtime)
    - Small extensible runtime core
    - Integreation of 2D/3D/Multimedia into user experience
    - Reliable and tractable
    - Minimal, reasonable core functionality
    - Behavior culling
    - Error reporting/handling
    - High performance
    - High fidelity

What X3D isn’t

- A language
- A stripped-down version of VRML
- A product
- One kind of product

X3DML

- From VRML node and field definitions to an Interface Definition Language (IDL)
- X3DML DTD (Document Type Definition)
- Provide content makers with more options in the tools (freeware, shareware, commercial products) with which they build complex content
- Differences VRML vs. X3DML
  - Metalanguage
  - XML Element Type Definition can explicitly define a tree of elements and the members of the tree
  - VMRL97 datatypes are included in VRML definition – X3DML only provides datatypes for the lexical description of the document
  - DEF/USE attributes of XML Element Type are explicitly defined in DTD
- XML schemas