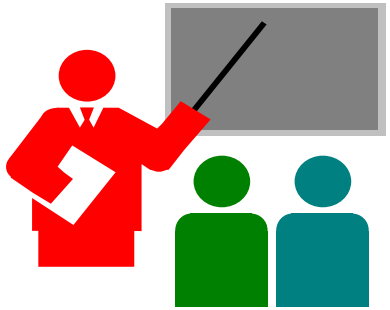
Virtual Intelligent Environments

Prof. Dr.-Ing. habil. Wolfgang Oertel
Faculty Information Technology / Mathematics
University of Applied Sciences Dresden

---

1. Introduction to Virtual Intelligent Environments
2. VRML: Concept and Working Environment
3. VRML: Syntax and Semantics
4. VRML: Geometric Objects and Transformations
5. VRML: Material, Illumination, and Observer
6. VRML: Animation and Interaction
7. VRML: Programming and Networking
8. VRML: Involvement of Multimedia
9. VRML: Intelligent Behaviour
10. Applications of Virtual Intelligent Environments

Demonstration VRML Practice VRML



Literatur

- VRML97: *The Virtual Reality Modelling Language*, VRML97. International Standard ISO/IEC 14772-1:1997 and ISO/IEC 14772-2:2004
- *Extensible 3D (X3D) encodings* ISO/IEC FDIS 19776-2. Web3D Consortium, Inc., 2004
- Walsh, A.; Bourges-Sevenier, M.: *Core Web3D*. Prentice Hall, Upper Seddle River, NJ, 2001
- Daly, L.; Brutzman, D.: *X3D - Extensible 3D Graphics for Web Authors*. Elsevier, London, 2007
- Quigley, E.: *JavaScrip*t. Pearson Education, Upper Saddle River, 2004

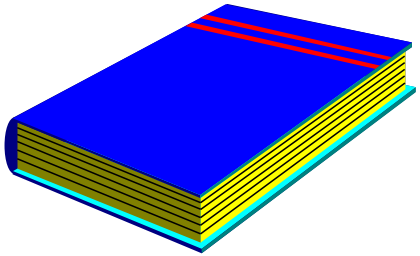
- Henning, A.: *Die andere Wirklichkeit: Virtual Reality* – Konzepte, Standards, Lösungen. Addison-Wesley, Bonn, 1997
- Hase, H.: Dynamische virtuelle Welten mit VRML 2.0: Einführung, Programme und Referenz. dpunkt, Heidelberg, 1997
- Kloss, J.; Rockwell, R.; Szabo, K.; Duchrow, M.: *VRML97: Der neue Standard für interaktive 3D-Welten im World Wide Web*. Addison-Wesley, Bonn, 1998
- Däßler, R.; Palm, H.: Virtuelle Informationsräume mit VRML. dpunkt, Heidelberg, 1998
- Burdea, G.; Coiffet, P.: *Virtual Reality Technology*. John Wiley & Sons, Hoboken, 2003

- Foley, J.; van Dam, A.; Feiner, S.; Hughes, J.: *Computer Graphics*. Addison-Wesley, Bonn, 1990
- Russell, S.; Norvig, P.: *Artificial Intelligence*. Pearson Education, München, 2004

- Web3D-Consortium: www.web3d.org
- WebReference: www.webreference.com
- IEEE Virtual Reality Conference (VR)

The course bases upon above literature.

Use of the script only for private purpose!



1. Introduction to Virtual Intelligent Environments

Prof. Dr.-Ing. habil. Wolfgang Oertel

Terms

Human Being

Ambient Intelligence — **Virtual Intelligent Environment**

???

Real Environment — **Virtual Environment**

Artificial Intelligence

Reality — **Virtual Reality**

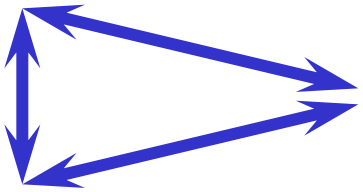
Motivation

10 reasons for the development of a new system generation:

World:

1. Four-dimensional existence of the world in space an time
2. Expansive material and energy processes in real environments
3. Many inaccessible and invisible real environments
4. Costs and dangers in real environments

Human:

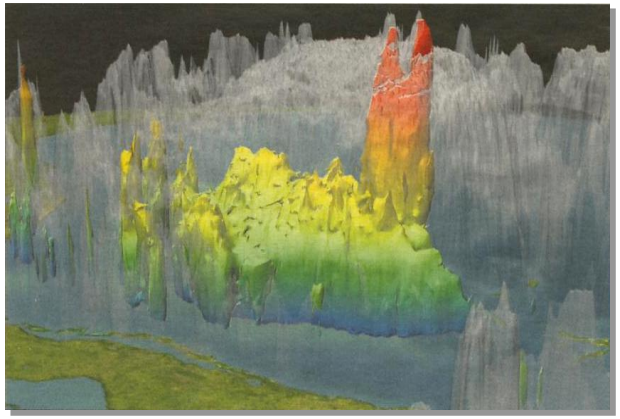


5. Human-specific sensors and actuators
6. Human-specific way of thinking
7. Intuitive access to world and computer

Computer:

7. Effective handling of more complex systems
8. Natural modeling of states and processes
9. Uniform world-wide communication medium
10. Intelligent behavior and interaction

Search for complete support
for all these tasks



Virtual Reality

Reality: Variety of states, processes, and regularities that exists outside and independent of the consciousness (of human being)

State: objects and relations (static, concrete, real))

Process: change of states (dynamic, concrete, real)

Regularity: general rules and laws (abstract, real)

Consciousness: highest form of the mapping of the reality (ideal)

Virtual Reality:

Medium that enables the human to **imagine** a complex computer-generated environment using several senses to **immerse** in this reality and to **interact** with it in order to permit the feeling to be present in it

Virtual: not real, apparent, pretended, seeming

Alternative Terms:

- Virtual Environment
- Augmented Virtuality
- Realistic Sensation
- Cyberspace
- Artificial Reality
- Augmented Reality
- Real Environment

VR

R

Alternative Definitions:

Consciously experienced complex illusion produced by data of all computers in the human brain. [Neuromancer, Gibson, 1984]

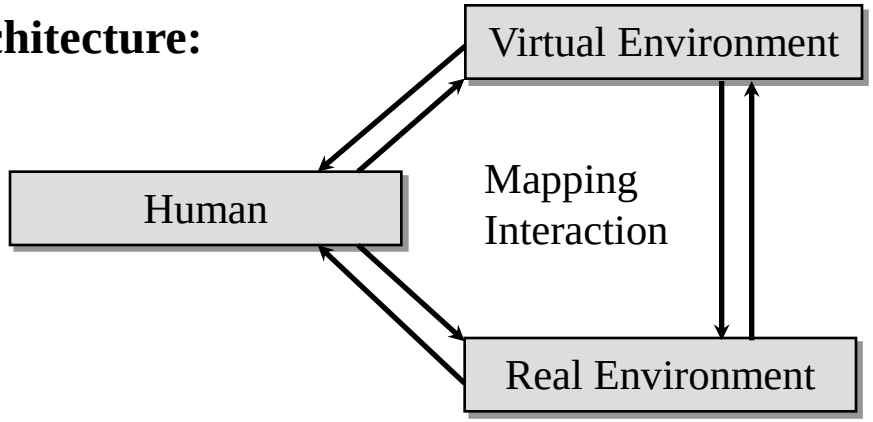
Animated audiovisually illustrated networked metaverse with communicating human-like avatars. [Snow Crash, Stephenson, 1990]

VR is shared and objectively present like the physical world, composable like a work of art, and as unlimited and harmless as a dream. [Jaron Lanier, VPL Texpo, 1989]

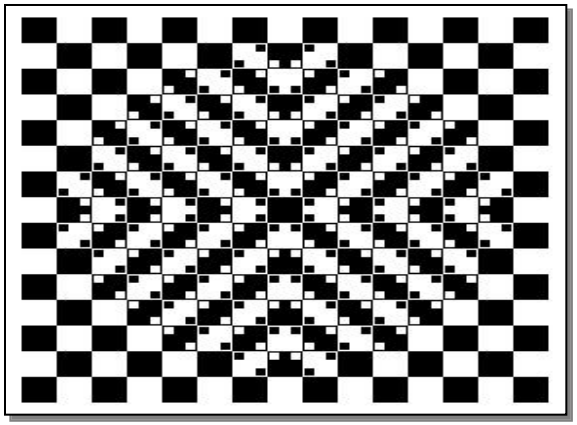
Complex Relationship:

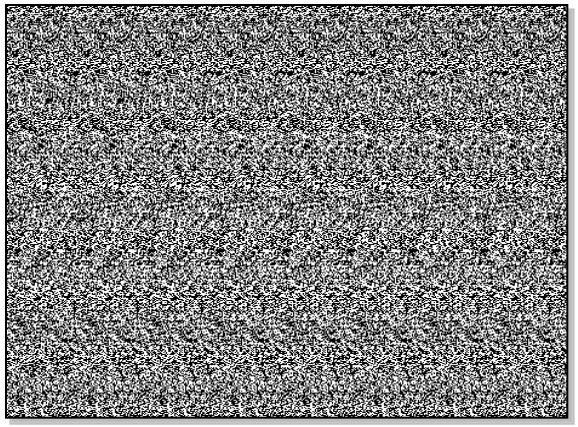
- between reality, human, and computer
- by sensors, actuators, several media
- and networking

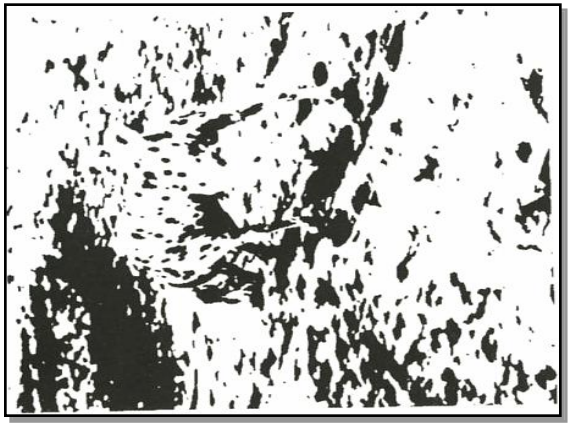
Architecture:



Immersion Test:







Basic Components

Computer Graphics: Processing of computer-internal spatiotemporal models and generation of images for presentation on external graphical devices

Spatiotemporal models:
- 2D
- 3D
- 4D

Graphic Operations:
- Drawing
- Rendering
- Animation

Multimedia: Computer-controlled integrated handling of several independent information coded in different media

Elementary Media:
- optic
- acoustic
- haptic
- kinesthetic
- gustatory, olfactory
- thermic

Complex Media:
- text
- graphics
- image
- video
- audio
- language

Operations:
- Perception
- Analysis
- Synthesis
- Presentation

Networking: Involvement and integration of several local and distributed resources

Local resources:
- file system
- libraries
- data bases

Distribute Resources
- computer networks
- world wide web
- wireless network

Artificial Intelligence

Intelligence: Set of human abilities, like knowledge storage, problem solving, planning, language processing, spatial navigation, pattern recognition;
→ no precise criterion

Operational criterion: Intelligence test, Turing test

Artificial Intelligence:
Medium with structures, operations, and behaviours occurring in natural **intelligent** systems

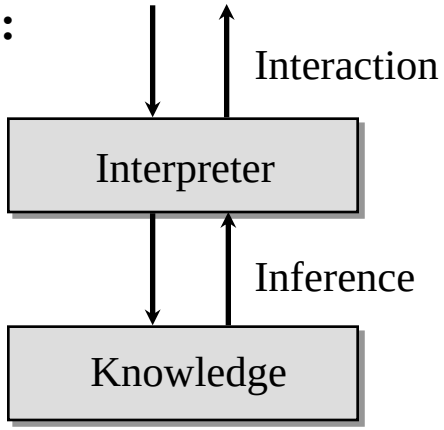
System categories:

Knowledge-based:
knowledge about reality is stored comprehensibly and used explicitly and consciously in inference processes
→ Expert in a special field

Behavior-based:
knowledge about reality is stored incomprehensibly and used implicitly and unconsciously in inference processes
→ Agent in everyday life

Knowledge representation:
Representation of information about the reality by a set of clauses (knowledge) and a mechanism to infer new clauses from existing ones (interpreter)

Architecture:

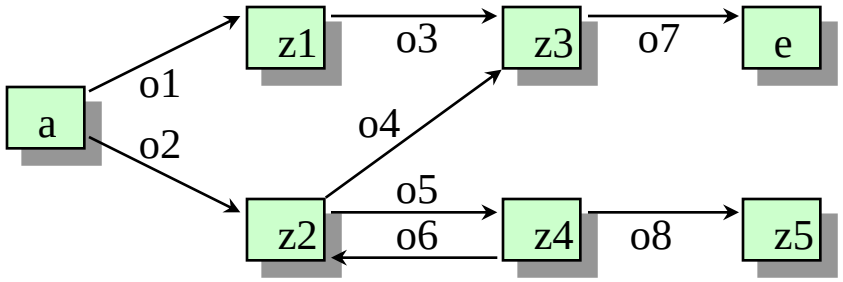


Virtual reality: Clause → Clause (Inference)

Real: Fact → Fact (Consequence)

Problem solving:
Search for a path in a state space connecting a start state with an end state by given operations

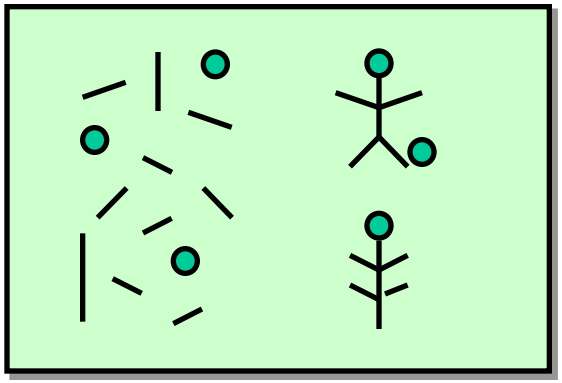
State graph:

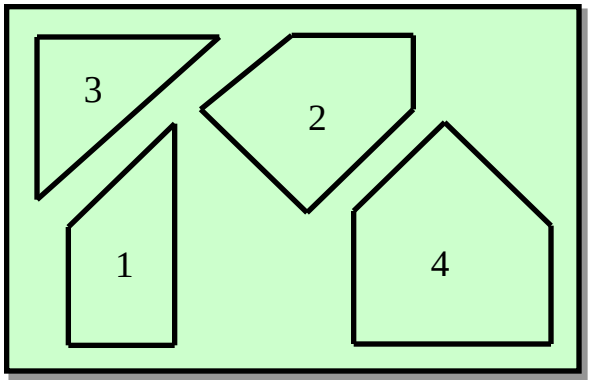


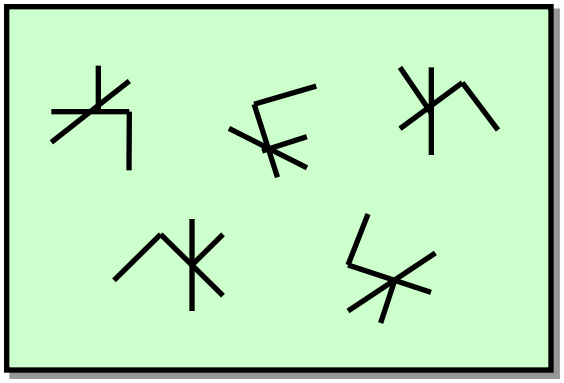
Search method:

Direction: forward, backward
Expansion: breadth first, depth first
Information: blind, heuristic
Memory: tree, net

Intelligence Test







Basic Components

Artificial Intelligence: Knowledge representation, problem solving, and behaviour structure

Knowledge representation:
- Logics
- Rules
- Frames
- Semantic Networks
- Constraints

Problem solving:
- Deduction
- Analogy
- Induction
- Search
- Optimisation
- Probabilistics
- Heuristics

Behaviour structure:
- Case bases
- Fuzzy systems
- Neural networks
- Genetic populations
- Social systems

Virtual Intelligent Environment

Motivation:
Combination of technologies of Virtual Reality and Artificial Intelligence

Virtual Intelligent Environment:
Virtual model of the surrounding real world with facilities realising selected intelligent functions of the human being

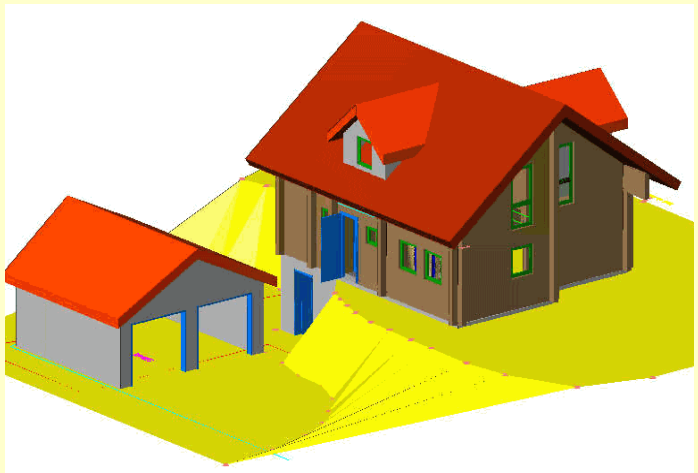
$$VR \supset VE \qquad VE \cup AI = VIE \qquad VIE \Leftrightarrow HB$$
$$R \supset RE \qquad RE \cup AI = AmI \qquad AmI \Leftrightarrow HB$$

$$R \Leftrightarrow VR$$
$$RE \Leftrightarrow VE$$
$$AmI \Leftrightarrow VIE$$

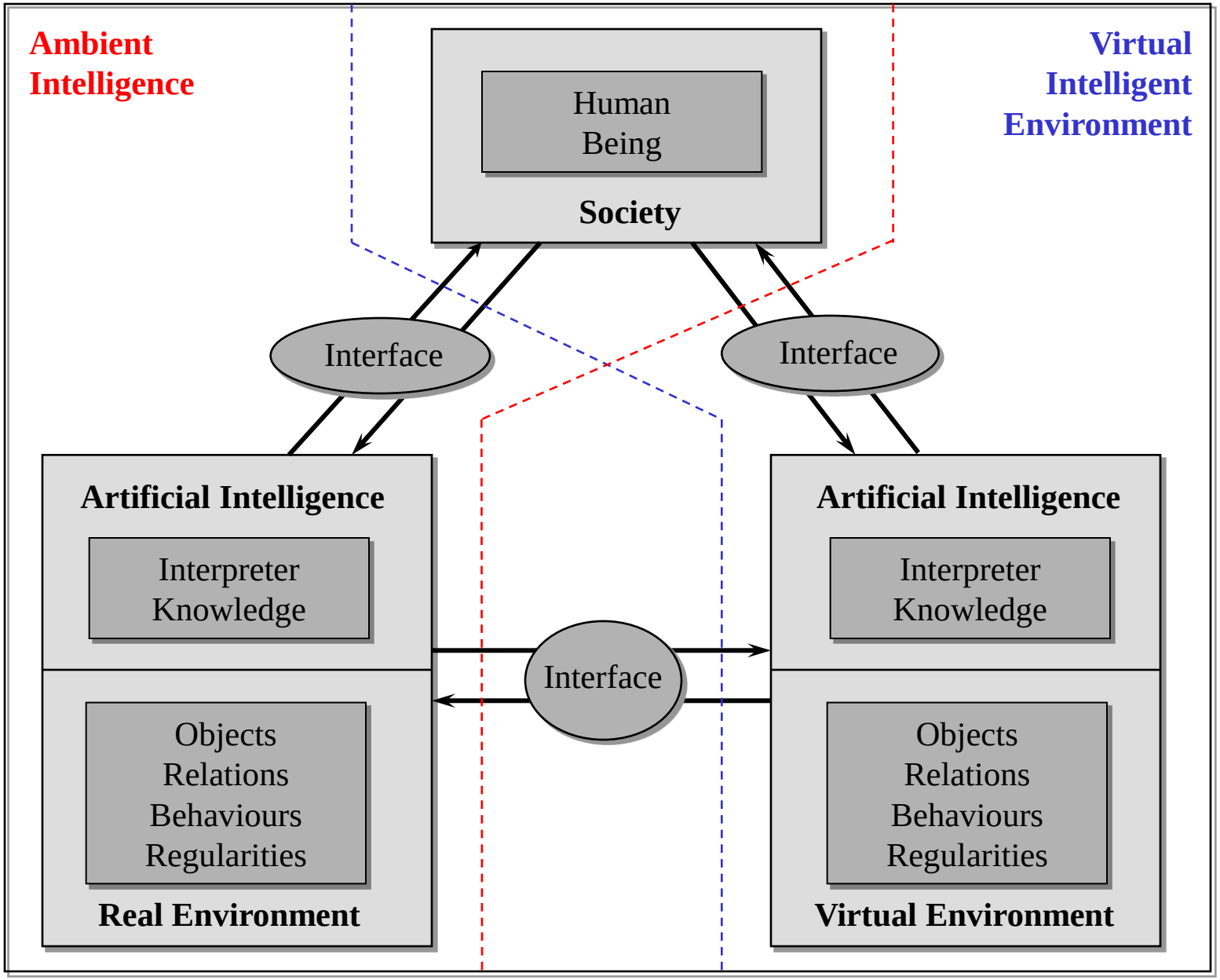
Application areas:

Virtual intelligent
- *building, clothing*
- *traffic, manufacturing, learning*
- *technical, scientific*

environment



System Architecture



Characteristics

- **Immersion:** Psycho-physiological involvement (sense to be in the real scene)
- **Interaction:** Interaction in both directions
- **Imagination:** Creation of images which are not really available
- **Intelligence:** Behaviour like a thinking human

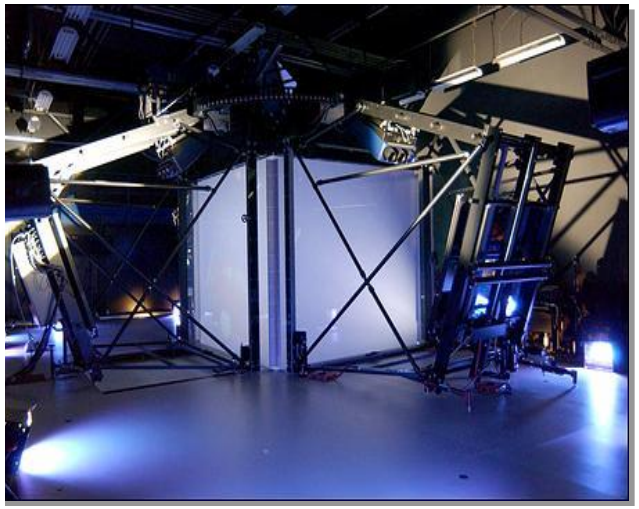
- **Modelling:** Mapping of real or artificial worlds
- **Statics / Dynamics:** Mapping of states and processes
- **Real time ability:** Temporal synchronisation with real processes
- **Navigation:** Motion through the scene
- **Multimedia:** Use of data of different media
- **Networking:** Overcoming of space and time by linking
- **Efficiency:** Small source requirements
- **Integration:** Cooperation of different basis components
- **Standardisation:** Uniformly defined languages and interfaces

→ **Cross sectional area:** New quality by combining several features

Hardware

- **Components:**

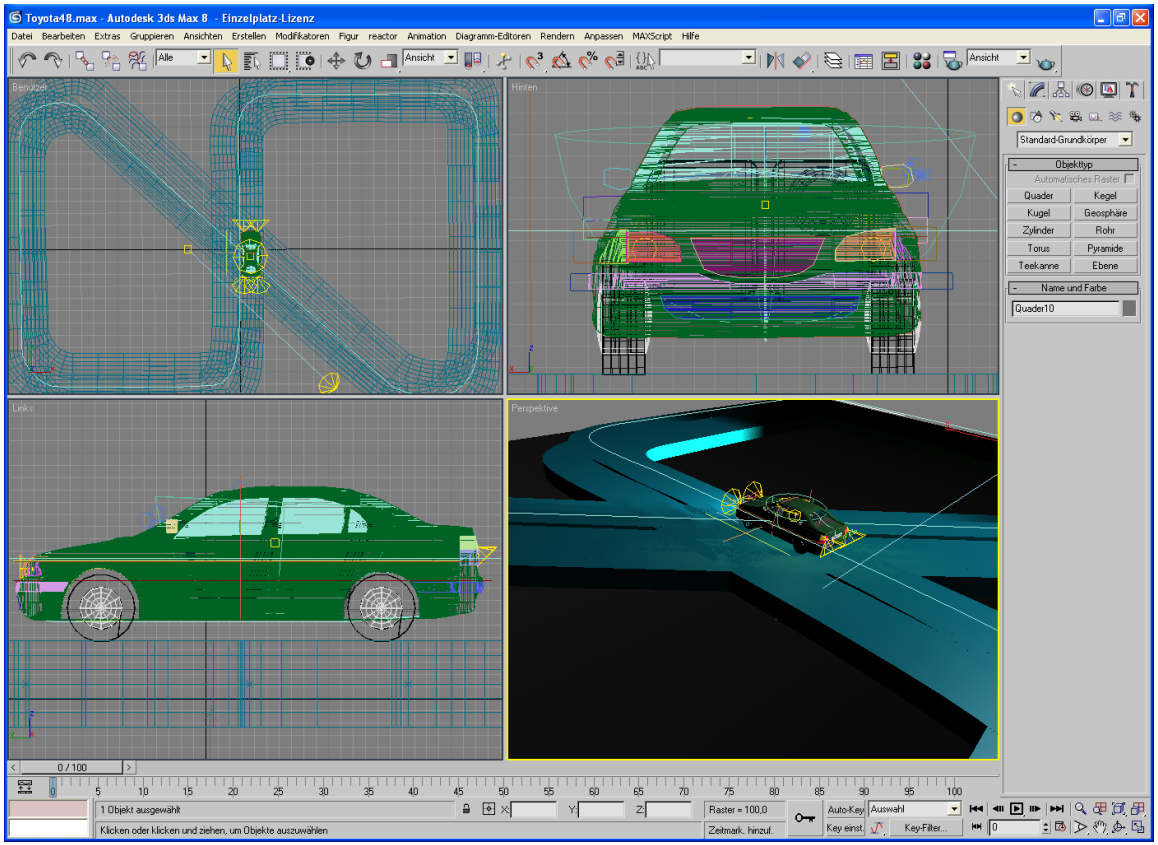
- Computer, Storage, Network
- Graphics Card, Periphery
- Printer, Plotter
- Display, Beamer
- Keyboard, Mouse, Tablet
- Camera, Scanner, Tracker
- Space ball, Digitizer
- Stereo display, Glasses
- Stereo beamer, Cave
- Head-mounted display
- Data glove, Data suit



Software

- **Concepts:**
  - Independent language
  - Library package
  - Language extension
  - Language translation
  - Separate Systems

- **Realizations:**
  - Basis: Operating system, database, network
  - Standards: OpenGL, DirectX, VRML, X3D, DXF, CommonLisp, ISOProlog
  - Systems: AutoCAD, 3DStudioMAX, IDL, VRML Viewers, CLisp, SWIProlog



Application Areas

- Construction and Design
- Production and Supervision
- Modeling and Simulation
- Mechanical engineering and Manufacturing
- Architecture and Civil engineering
- Geoinformatics, Cartography
- Electrical engineering
- Medicine
- Commercial and Business graphics
- Scientific-technological visualization
- Human-machine communication
- Media technology
- Art
- Entertainment and Games
- Computer animation, Virtual Reality

Scientific, Technical Orientation

Medial, Entertainment Orientation

Categories:
- technical
- scientific
- institutional
- economic
- private

Effect: overall social dimension

History

Prehistory: non-computer-based mapping and modelling of the reality

1941: Computer as new technical basis
1950: Development of hardware and software basis
1960: Computer graphics, artificial intelligence concepts

1980: Computer graphics, artificial intelligence dissemination
1984: Graphics, artificial intelligence language standards
1984: Cyberspace [William Gibson]
1989: Virtual reality [Jaron Lanier]
1990: Networking, multimedia, 3D graphics

1997: VRML standardisation (ISO)
1998: Ambient intelligence (Philips)
2004: X3D standardisation (ISO)

Presence: Large number of systems and projects

Future: Virtual Intelligent Environment
as new system generation with social component !?
Anyway: huge development impulse

2. VRML: Concept and Working Environment

Prof. Dr.-Ing. habil. Wolfgang Oertel

Claim

3 paradigms of information technology:

1. **Mainframe computer:** Text, command, report, reading, typing, technician → **institutional**
2. **Desktop computer:** 2D graphics, window, point, click, brain worker → **personal**
3. **Internet computer:** Multimedia, 3D graphics, scene, avatar, walking, chat, everybody → **social**

→ IT-Paradigm 3 needs a **new language generation:**

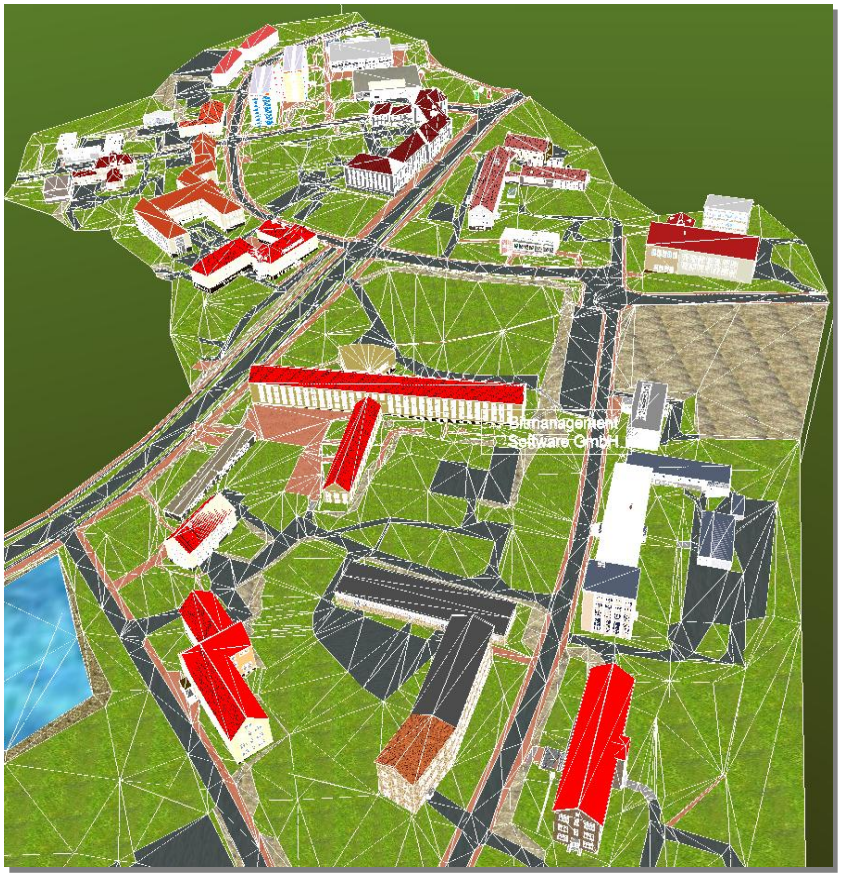
Features:

- learnable (for everybody)
- efficient (for real applications)
- executable (on each computer platform)
- communicable (between several systems)

Social Aspect

- VRML is a technical, but also a social experiment.
- People together want to be creatively active in a recognizable world.
- People range in the computer-generated word as in their familiar real world.
- Everybody contributes to the construction of the virtual world.

- VRML is not defined definitely by experts.
- VRML is developing by the work of people using the language.



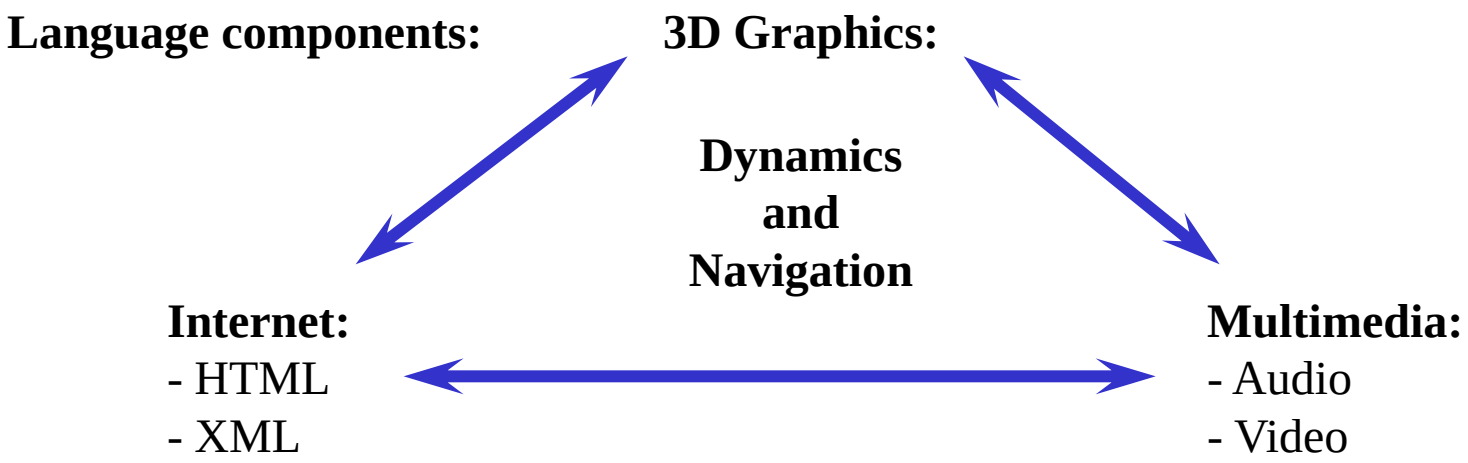
Community, social environment

Unity of:

- Language development and
- World development

Language Structure

VRML (Virtual Reality Modeling Language) uses 3D computer graphics to navigate in and interact with a changing spatial virtual world of multimedia objects realized on computers with sensors and actuators distributed over the internet.



Relationship:	- between already existing language concepts - by integration with defined interfaces

Features:	- Statics, dynamics	- Modeling, navigation
	- real time ability, efficiency	- Integration, standards

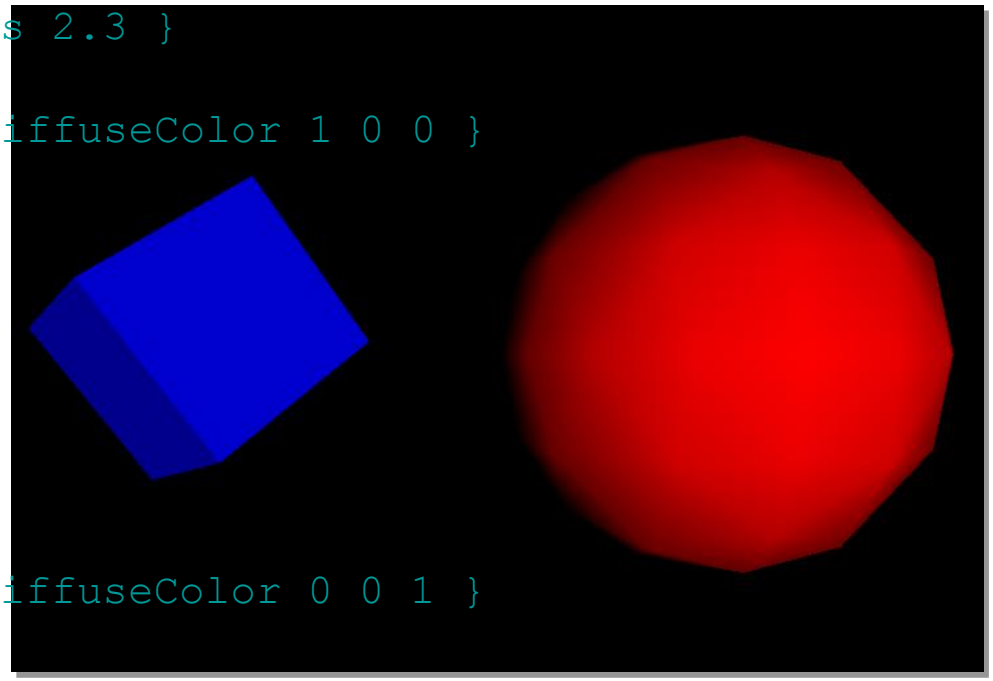
VRML Components

- VRML is a declarative language.

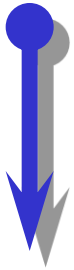
- A virtual world consists of nodes.
- Several nodes build a scene.
- Nodes are organized in a hierarchical structure.
- The user view consists of view point, direction, and angle.
- The user walks, flies or jumps through the scene.
- An animation changes objects of a scene.
- Events can cause changes.
- Sensors allow to interact with objects.
- Images, Textures, Audios and Videos can be embedded.
- Hyperlinks connect objects with other objects or scenes in the Web.

VRML Code Example

```
#VRML V2.0 utf8
Transform { # begin of world
  children [
    NavigationInfo { headlight FALSE }
    DirectionalLight { # illuminating light
        direction 0 0 -1
    }
    Transform { # red sphere
      translation 3 0 1
      children [
        Shape {
          geometry Sphere { radius 2.3 }
          appearance Appearance {
            material Material { diffuseColor 1 0 0 }
         }}]}
    Transform { # blue box
      translation -2.4 0.2 1
      rotation     0 1 1 0.9
      children [
        Shape {
          geometry Box {}
          appearance Appearance {
            material Material { diffuseColor 0 0 1 }
        }}]}] # end of world }
```



Language Versions



VRML1.0: 1994: 1. and 2. international WWW Congress

VRML2.0: 1996: SIGGRAPH Congress, VRML Consortium

VRML97: 1997: ISO Standardization

X3D: 2004: ISO Standardization, Web3D Consortium

Differences:

VRML1.0:

- link to HTML
- file format file.wrl
- static world
- no multimedia
- no Scripts
- no 2D objects
- polygonal objects
- no networking
- for special computers

VRML2.0 / 97:

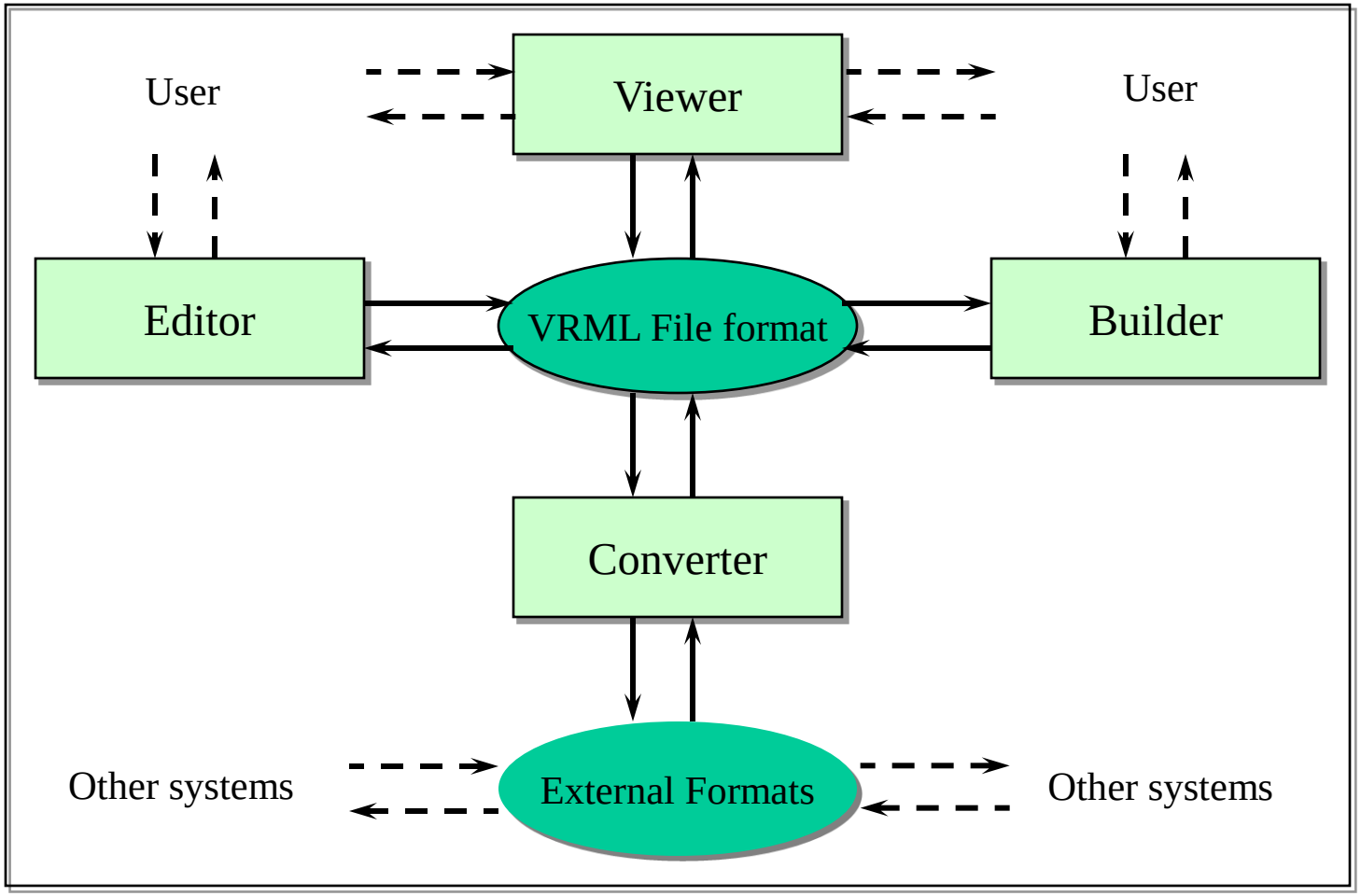
- link to HTML
- file format file.wrl
- dynamic world
- multimedia
- Scripts
- no 2D objects
- polygonal objects
- no networking
- for special computers

X3D:

- embedding in XML
- file format file.x3d
- dynamic world
- multimedia
- Scripts
- 2D objects
- polynomial objects
- networking
- for all computers

Working Environment

VRML is a **file format** (xxx.wrl).
VRML can be manipulated by a **text editor** or a **builder**.
VRML can be activated by a **Viewer** (or Browser).
VRML can be transferred in other languages by a **Converter**.



VRML Viewer

Components:

- **Walk:** Walk with gravity and collision detection through the space
- **Fly:** Move arbitrarily through the space
- **Study:** Observe objects from different points of view

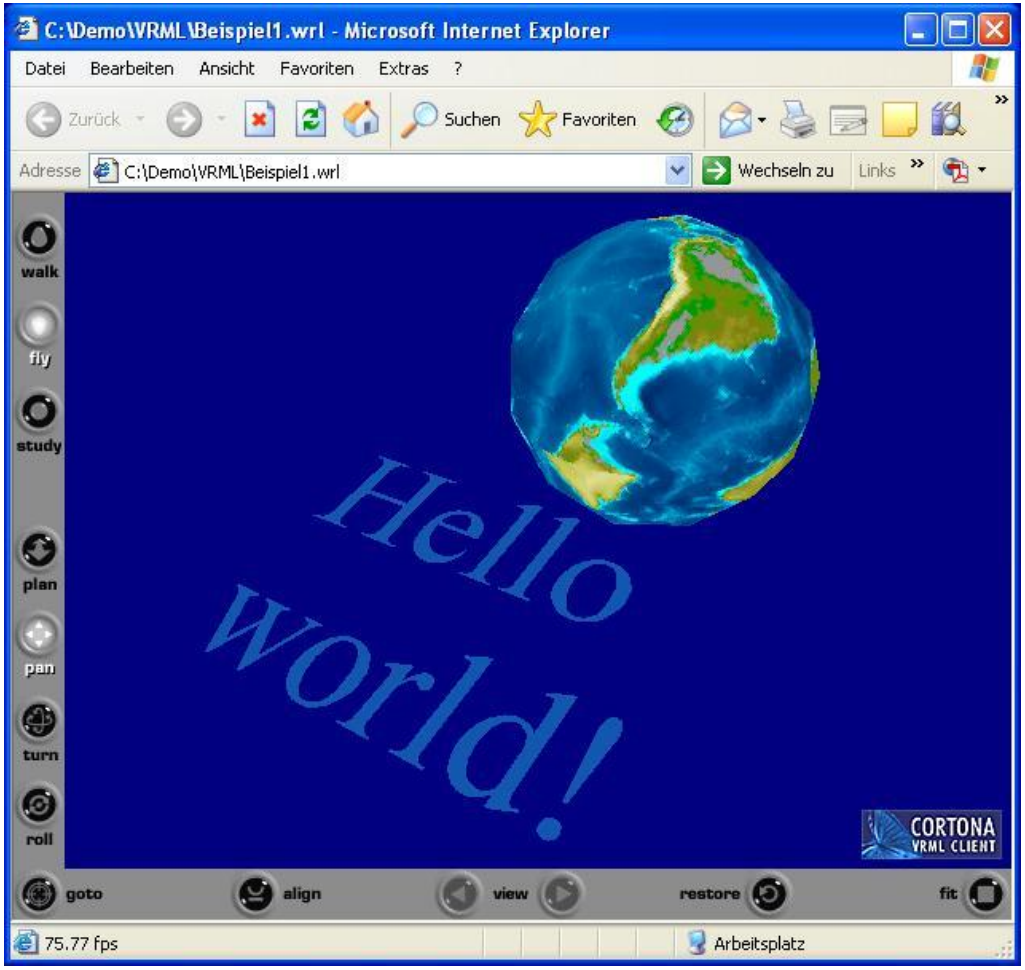
- **Plan:** forward, yaw
- **Pan:** up, left
- **Turn:** pitch, yaw
- **Roll:** pitch, roll

- **Goto:** got to an object
- **Align:** align the camera horizontally
- **View:** go to a view point
- **Restore:** go to the initial point
- **Fit:** view the entire scene

- **Speed:** determine navigation speed
- **Avatar:** determine avatar size
- **Headlight:** switch illumination ahead

VRML-Browser:

- Cortona VRML Client (ParallelGraphics)
- Cosmoplayer (Cosmosoftware)
- Octagon Free Player (Octaga)
- Contact VRML/X3D (Bitmanagement)
- Instantreality (Fraunhofer IGD)



ParallelGraphics: Croton VRML Client, User Intercace, 2005

VRML References

Books: - …

Standards:
- VRML: ISO/IEC 14772:1997
- X3D: ISO/IEC 19775:2004

General Websites:
- Web3D Consortium: www.web3d.org
- WebReference: www.webreference.com

VRML-Specifications:
- www.web3d.org/x3d/specifications
- www.graphcomp.com/info/specs/vrml

VRML-Viewers:
- FreeWrl: freewrl.sourceforge.net
- Xj3D: www.xj3d.org
- OpenVRML: openvrml.sourceforge.net

VRML Viewer-Plugins:
- Cosmo Player: www.karmanaut.com/cosmo/player
- Cortona VRML Client: www.parallelgraphics.com
- Blaxxun Contact: www.blaxxun.de
- BS Contact VRML/X3D: www.bitmanagement.de
- Octaga: www.octaga.com
- Instantreality: www.instant-reality.com

VRML Editors:
- VrmlPad: www.parallelgraphics.com/products/vrmlPad
- Rendersoft VRML Editor: www.homer.pacific.net.sg/~jupboo

3. VRML: Syntax and Semantics

Prof. Dr.-Ing. habil. Wolfgang Oertel

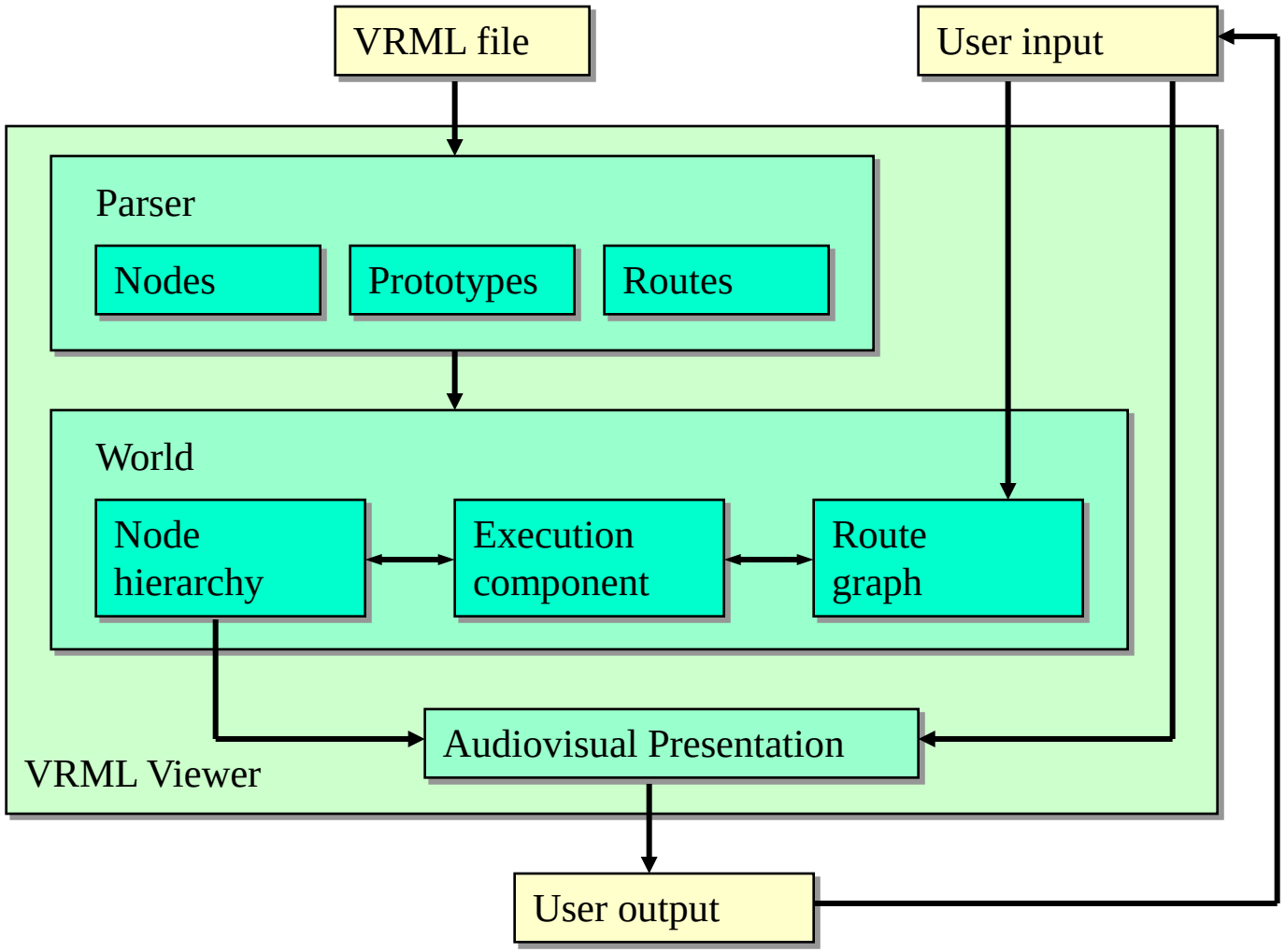
Syntax and Semantics

defines - structure of VRML components and their meaning during the interpretation:

- lexical elements
- data types
- value ranges
- coordinate systems
- syntactic components
- semantic effects

Set of syntactic and semantic conventions

Conceptual Viewer Model



Language Syntax

Header: Identification and Coding

#VRML V2.0 utf8 [*comment*] *lineterminator*

Scene description: Description of the World

vrmlScene ::= *statements* ;
statements ::= *statement* | *statement statements* | *empty* ;
statement ::= *nodeStatement* | *protoStatement* | *routeStatement* ;

Nodes: Hierarchical organized objects and their features

nodeStatement ::= *node* | **DEF** *nodeNameId node* | **USE** *nodeNameId* ;
node ::= *nodeTypeId* **{** *nodeBody* **}** | **Script {** *scriptBody* **}** ;

Prototypes: Definition of new node types

protoStatement ::= *proto* | *externproto* ;
proto ::= **PROTO** *nodeTypeId* **[** *interfaceDeclarations* **] {** *protoBody* **}** ;
externproto ::= **EXTERNPROTO** *nodeTypeId* **[** *externInterfaceDeclarations* **]** *URLList* ;

Routes: Propagation of events between nodes

routeStatement ::= **ROUTE** *nodeNameId* **.** *eventOutId* **TO** *nodeNameId* **.** *eventInId* ;

Node body: Specification of node components

nodeBody ::= *nodeBodyElement* | *nodeBodyElement nodeBody* | *empty* ;
scriptBody ::= *scriptBodyElement* | *scriptBodyElement scriptBody* | *empty* ;
nodeBodyElement ::= *fieldId fieldValue* | *fieldId* **IS** *fieldId* | *eventInId* **IS** *eventInId* |
eventOutId **IS** *eventOutId* | *routeStatement* | *protoStatement* ;

Fields: Specification of node features

fieldType ::= **MFColor** | **MFFloat** | **MFInt32** | **MFNode** | **MFRotation** | **MFString** |
MFTime | **MFVec2f** | **MFVec3f** | **SFBool** | **SFColor** | **SFFloat** | **SFImage** |
SFInt32 | **SFNode** | **SFRotation** | **SFString** | **SFTime** | **SFVec2f** | **SFVec3f** ;
sfnodeValue ::= nodeStatement | ***NULL*** ;
mfnodeValue ::= *nodeStatement* | **[ ]** | **[** *nodeStatements* **]** ;

General rules:

- Terminal symbols: ., {, }, [,]
- Key words: DEF, EXTERNPROTO, FALSE, IS, NULL, PROTO, ROUTE, TO, TRUE, USE, eventIn, eventOut, exposedField, field
- Separators: carriage return, line feed, space, tab, comma,
- Comments: #
- Strings: "…"
- Abrogation of symbol semantics: \
- Names: Case sensitivity, no control symbols, no beginning with number, +, or -

Field Descriptions

Field types:

- field:	static field (not changeable)	
- exposedField:	dynamic field (changeable)	
- eventIn:	event reception field	
- eventOut:	event creation field	Examples:
- single-valued field:	field with one value	foo 1
- multiple-valued field:	field with several values	foo [1 2 3 4]

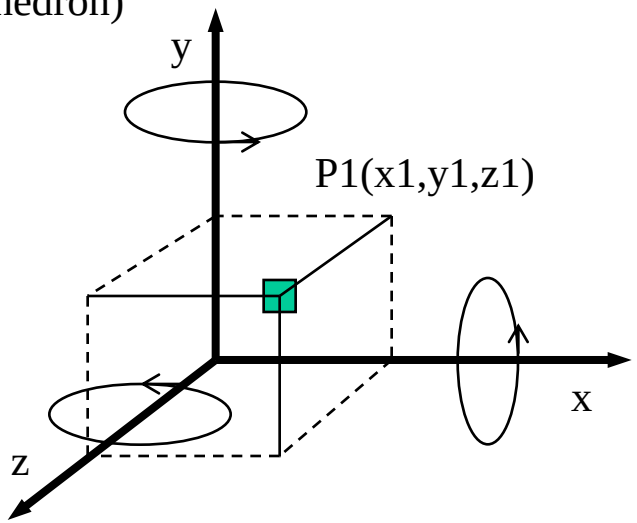
Data types:	Examples:
- **SFBool:**	fooBool FALSE
- **SFColor and MFColor:**	fooColor [1.0 0. 0.0, 0 1 0, 0 0 1]
- **SFFloat and MFFloat:**	fooFloat [3.1415926, 12.5e-3, .0001]
- **SFImage:**	fooImage 2 4 3 0xFF0000 0xFF00 0 0 0 0 0xFFFFFF 0xFFFF00
- **SFInt32 and MFInt32:**	fooInt32 [17, -0xE20, -518820]
- **SFRotation and MFRotation:**	fooRot 0.0 1.0 0.0 3.14159265
- **SFString and MFString:**	fooString ["One, Two, Three", "He said, \"She did it!\""]
- **SFTime and MFTime:**	fooTime 0.0
- **SFVec2f and MFVec2f:**	fooVec2f [42 666, 7 94]
- **SFVec3f and MFVec3f:**	fooVec3f [1 42 666, 7 94 0]
- **SFNode and MFNode:**	fooNode [Transform { translation 1 0 0 } DEF CUBE Box { } USE CUBE USE]

Ranges and Units

Value ranges:
correspond to the those of usual programming languages (int, float, bool, ...)

Name spaces:
Each VRML file has its own name space at run time (with root nodes and their successors).
Extensions are Prototype instances and Inline nodes.

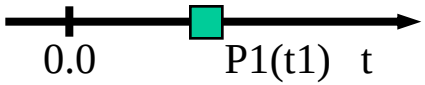
Space coordinate system:
(axis trihedron)



Units

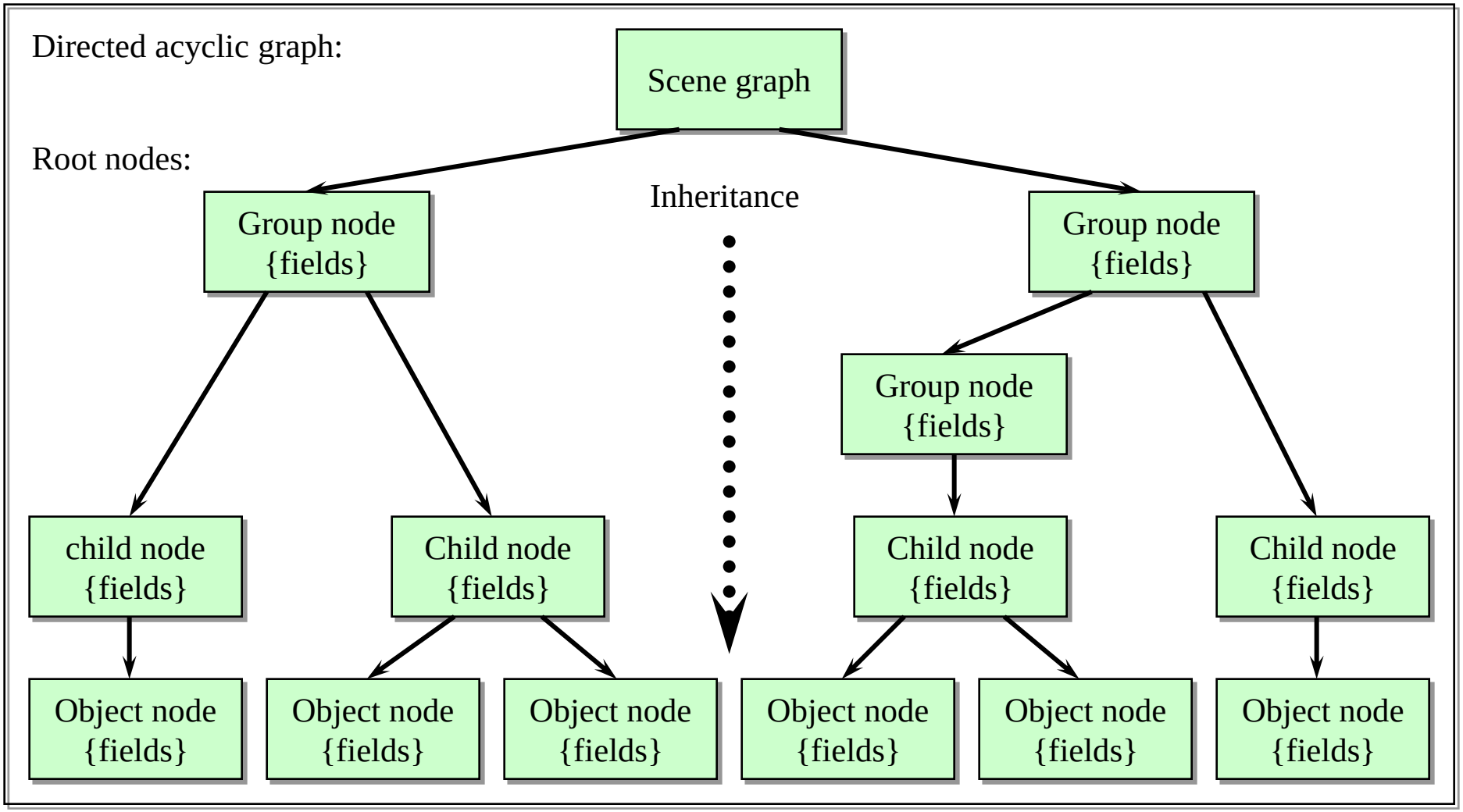
Category	Unit
Distance	Meter
Angle	Radian
Time	Second
Color	RGB ([0,1], [0,1], [0, 1])

Time coordinate system:
(axis beam)



Time (0.0) is equivalent to
00:00:00 GMT January 1, 1970

Scene Graph Structure



- Group node: Collection of child or group nodes with inheritable features
- Child node: Collection of object nodes
- Object node: Elementary node

Node Categories

Goup nodes:

- Anchor
- Billboard
- Collision
- Group
- Inline
- LOD
- Switch
- Transform

Child nodes:

- All group nodes
- Background
- ColorInterpolator
- CoordinateInterpolator
- CylinderSensor
- DirectionalLight
- Fog
- FontStyle
- NavigationInfo
- NormalInterpolator
- OrientationInterpolator
- PlaneSensor
- PointLight
- PositionInterpolator
- ProximitySensor
- ScalarInterpolator
- Script
- Shape
- Sound
- SpotLight
- SphereSensor
- TimeSensor
- TouchSensor
- Viewpoint
- VisibilitySensor
- WorldInfo

Object nodes:

- Appearance
- AudioClip
- Box
- Color
- Cone
- Coordinate
- Cylinder
- ElevationGrid
- Extrusion
- ImageTexture
- IndexedFaceSet
- IndexedLineSet
- Material
- MovieTexture
- Normal
- PixelTexture
- PointSet
- Sphere
- Text
- TextureCoordinate
- TextureTransform

Language Semantics

Group and Collision Nodes:

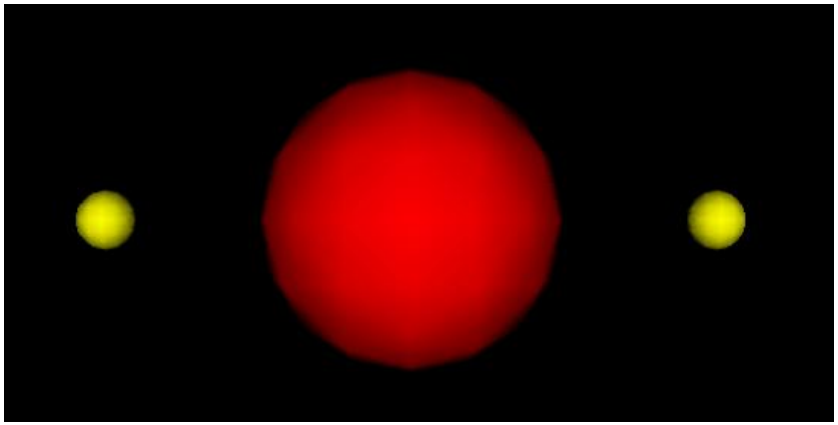
```
Group {
  eventIn       MFNode    addChildren
  eventIn       MFNode    removeChildren
  exposedField  MFNode    children []
  field         SFVec3f   bboxCenter 0 0 0     # (- ∞, ∞)
  field         SFVec3f   bboxSize -1 -1 –1    # (0, ∞) or -1,-1,-1 }
```

```
Collision {
  eventIn       MFNode    addChildren
  eventIn       MFNode    removeChildren
  exposedField  MFNode    children []
  exposedField  SFBool    collide TRUE
  field         SFVec3f   bboxCenter 0 0 0     # (- ∞, ∞)
  field         SFVec3f   bboxSize -1 -1 -1    # (0, ∞ ) or -1,-1,-1
  field         SFNode    proxy NULL
  eventOut      SFTime    collideTime
```

Bounding box: Cuboid whose edges are aligned parallel to the axes x, y, z and have the lengths of the respective maximal distances of the enclosed objects

Code-Example

```
#VRML V2.0 utf8
Group {
 children [
  DirectionalLight {direction 0 0 -1}
  Collision {collide FALSE
   children [Shape {
    geometry Sphere {}
    appearance Appearance {
     material Material {diffuseColor 1 0 0}}}]}
  Collision {collide TRUE
   children [
    Transform {
     translation 2 0 0
     children DEF Ball Shape {
      geometry Sphere {radius .2}
      appearance Appearance {
       material Material {
        diffuseColor 1 1 0}}}}]}
  Collision { collide TRUE
   children [
    Transform {
     translation -2 0 0
     children USE Ball}]}]}
```



4. VRML: Geometric Objects and Transformations

Prof. Dr.-Ing. habil. Wolfgang Oertel

Elementary geometric Objects are defined:
• by specified or default values of size and shape
• by specified or default position and orientation
• in the global world coordinate system

Complex geometric Objects are created:
• by collection of defined objects
• in a group node

Spatially transformed geometric Objects are created:
• by scale, rotation, and translation in a transform node

A **Transform node** defines:

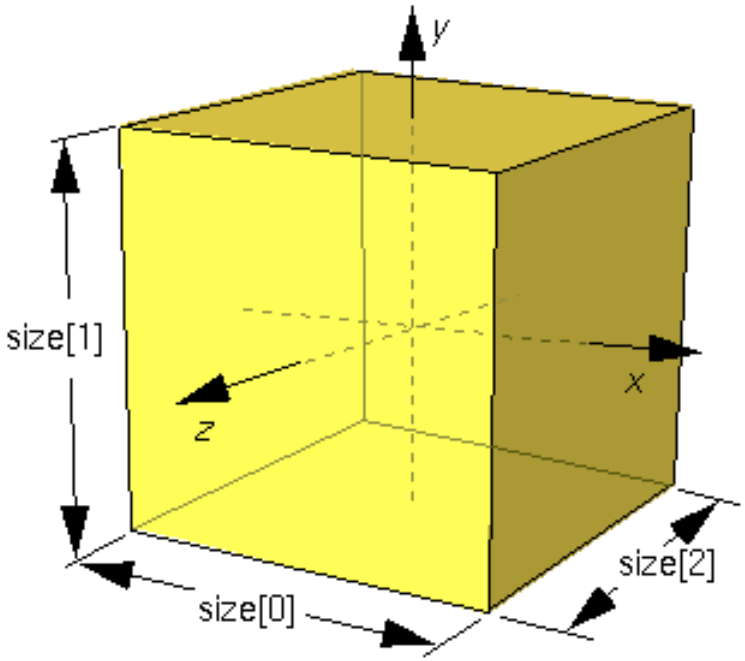
• an own local coordinate system
• for the contained child nodes
• relatively to the superordinated coordinate system

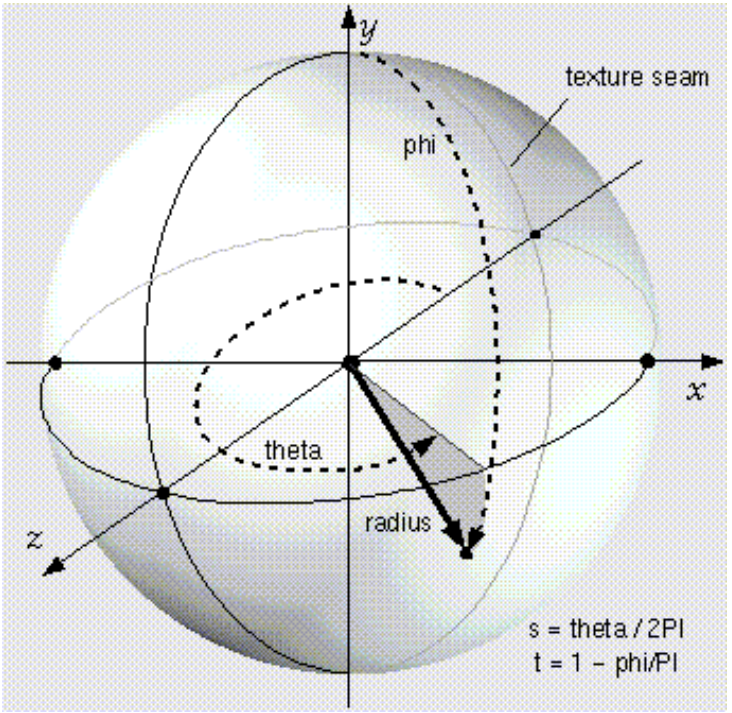
Hierarchy of coordinate systems with bottom-up accumulated transformations

Elementary Geometric Objects

Box { field SFVec3f **size 2 2 2** # (0, ∞) **}**

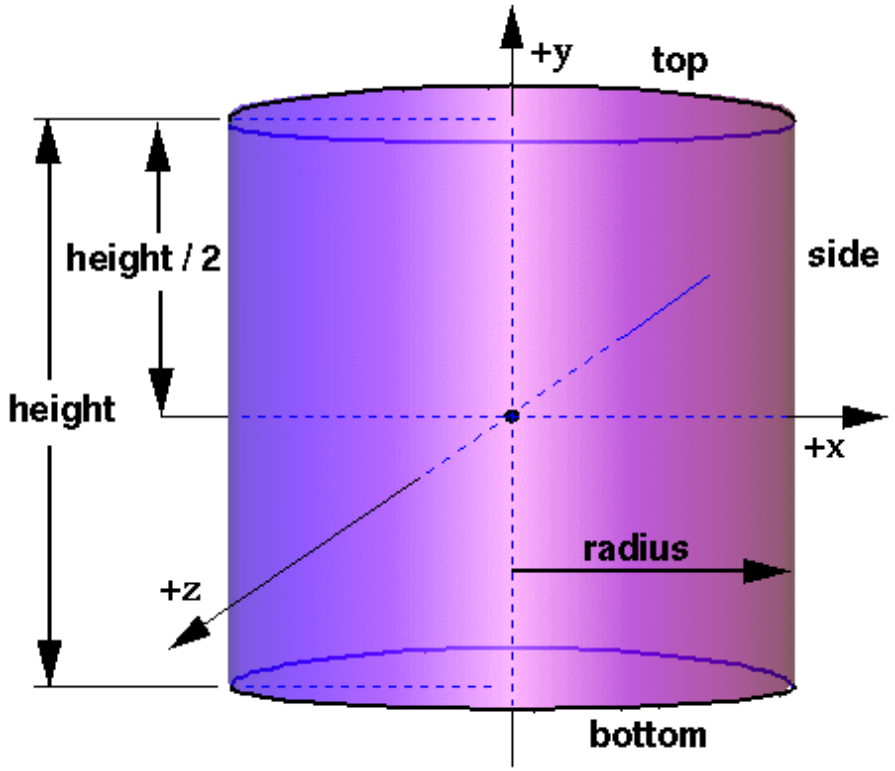
Sphere { field SFFloat **radius 1** # (0, ∞) **}**

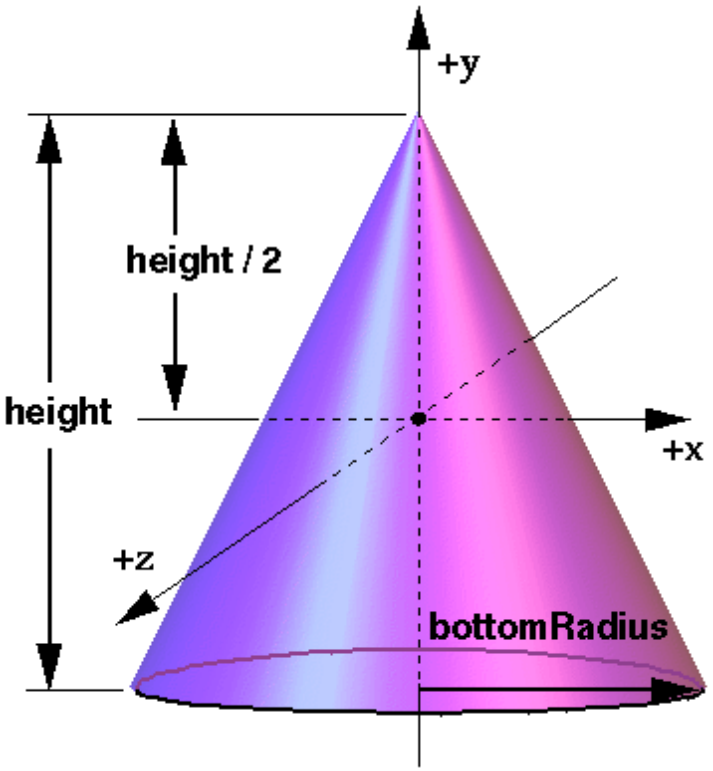




Cylinder { field SFBool **bottom TRUE**
field SFFloat **height 2** # (0, ∞)
field SFFloat **radius 1** # (0, ∞)
field SFBool **side TRUE**
field SFBool **top TRUE }**

Cone { field SFFloat **bottomRadius 1** # (0, ∞)
field SFFloat **height 2** # (0, ∞)
field SFBool **side TRUE**
field SFBool **bottom TRUE** }

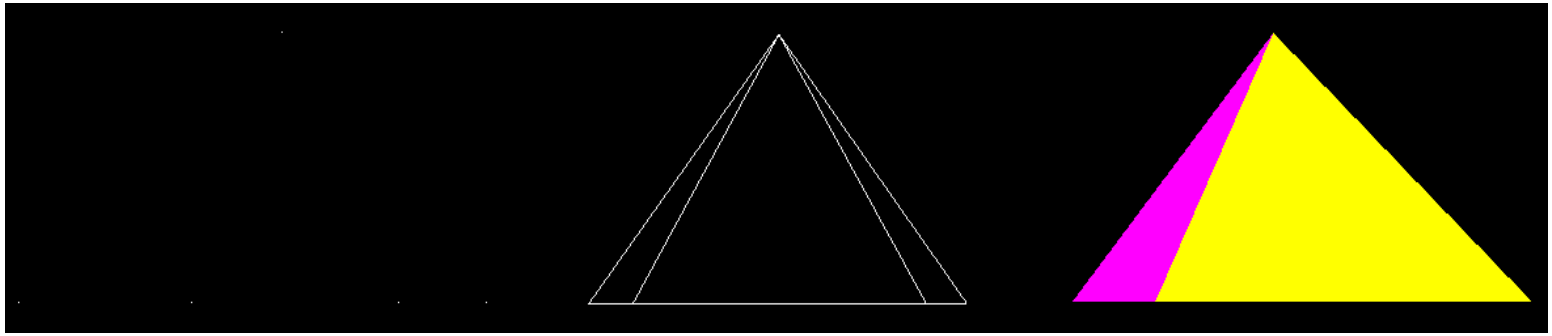




PointSet {
exposedField SFNode **color NULL**
exposedField SFNode **coord NULL }**

IndexedLineSet {
eventIn MFInt32 set_colorIndex
eventIn MFInt32 set_coordIndex
exposedField SFNode **color NULL**
exposedField SFNode **coord NULL**
field MFInt32 **colorIndex []** # [-1,∞)
field SFBool **colorPerVertex TRUE**
field MFInt32 **coordIndex []** # [-1,∞) **}**

IndexedFaceSet {
eventIn MFInt32 set_colorIndex
eventIn MFInt32 set_coordIndex
eventIn MFInt32 set_normalIndex
eventIn MFInt32 set_texCoordIndex
exposedField SFNode **color NULL**
exposedField SFNode **coord NULL**
exposedField SFNode **normal NULL**
exposedField SFNode **texCoord NULL**
field SFBool **ccw TRUE**
field MFInt32 **colorIndex []** # [-1, ∞)
field SFBool **colorPerVertex TRUE**
field SFBool **convex TRUE**
field MFInt32 **coordIndex []** # [-1,∞)
field SFFloat **creaseAngle 0** # [0, ∞)
field MFInt32 **normalIndex []** # [-1,∞)
field SFBool **normalPerVertex TRUE**
field SFBool **solid TRUE**
field MFInt32 **texCoordIndex []** # [-1, ∞) **}**



General usable nodes:

Coordinate {
exposedField MFVec3f **point []** # (-∞ ,∞) **}**

Normal {
exposedField MFVec3f **vector []** # (-∞ ,∞) **}**

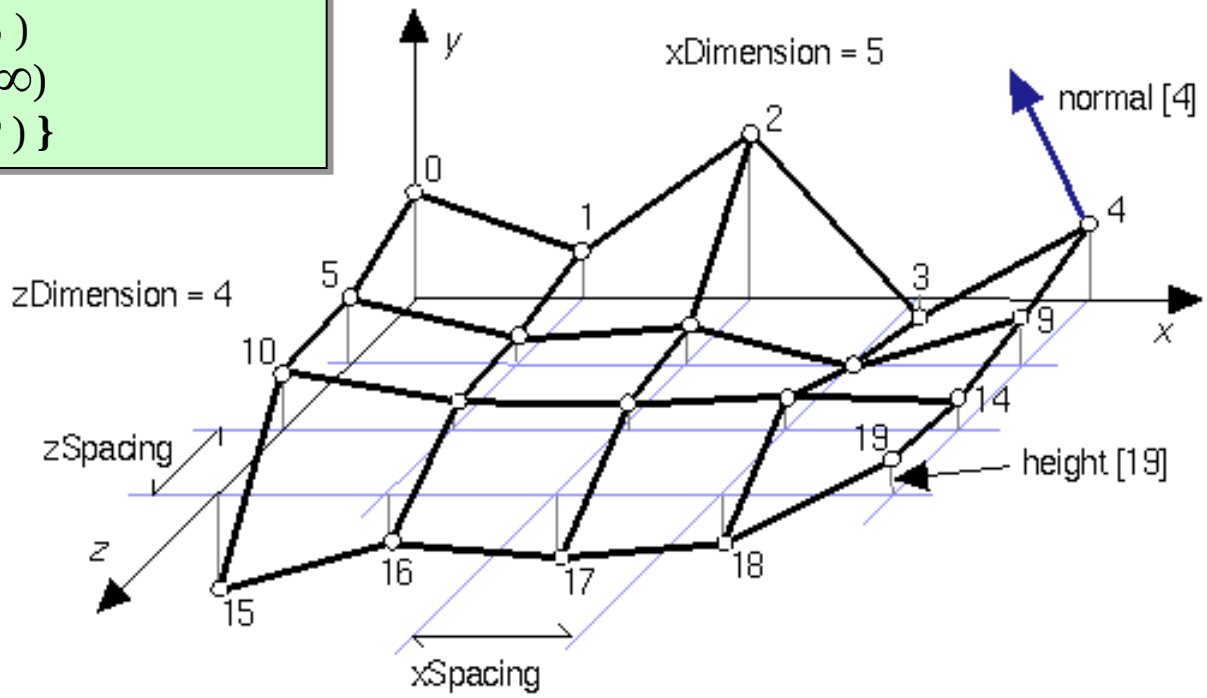
Color {
exposedField MFColor **color []** # [0 , 1] **}**

TextureCoordinate { exposedField MFVec2f **point []** # (- ∞, ∞) **}**

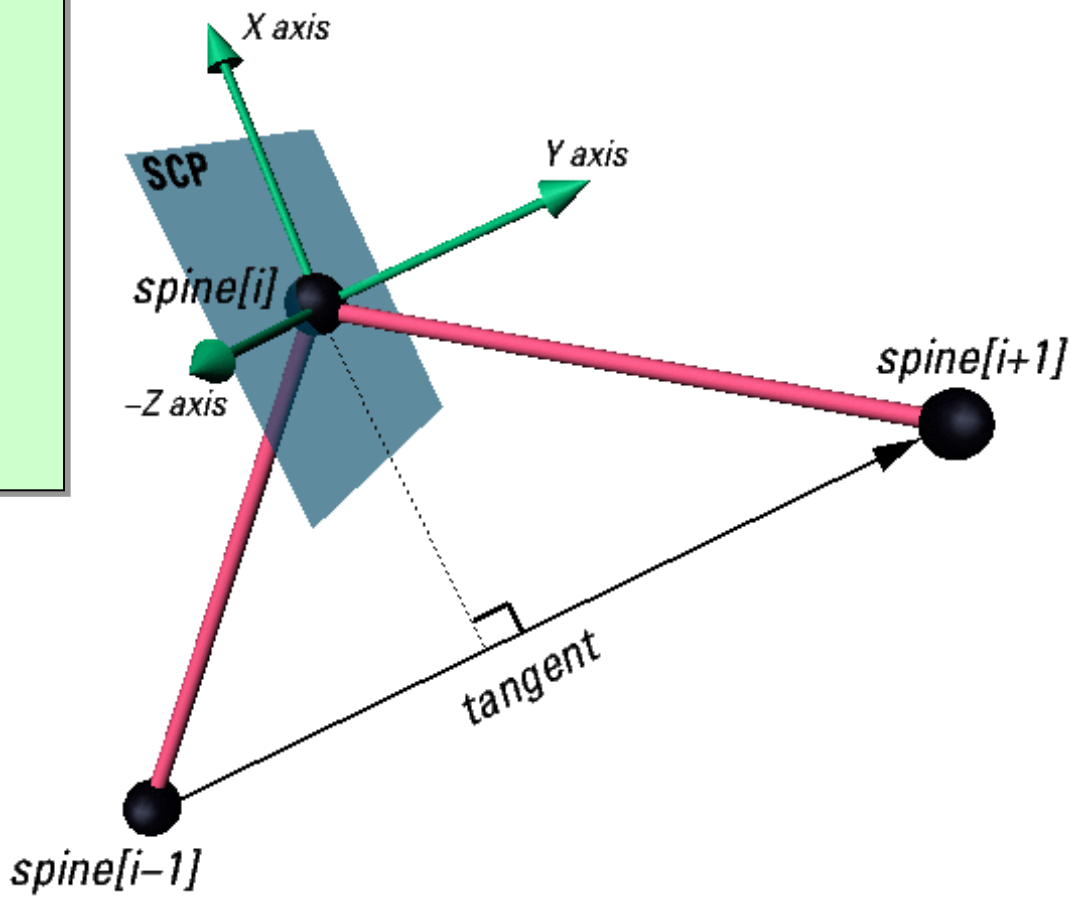
Shape as combination of geometry and appearance:

Shape { exposedField SFNode **appearance NULL**
exposedField SFNode **geometry NULL }**

ElevationGrid {
  eventIn MFFloat set_height
  exposedField SFNode **color NULL**
  exposedField SFNode **normal NULL**
  exposedField SFNode **texCoord NULL**
  field MFFloat **height [] #** (-∞ , ∞)
  field SFBool **ccw TRUE**
  field SFBool **colorPerVertex TRUE**
  field SFFloat **creaseAngle 0** # [0, ∞]
  field SFBool **normalPerVertex TRUE**
  field SFBool **solid TRUE**
  field SFInt32 **xDimension 0** # [0, ∞)
  field SFFloat **xSpacing 1.0 #** (0,∞)
  field SFInt32 **zDimension 0** # [0, ∞)
  field SFFloat **zSpacing 1.0 #** (0,∞) **}**

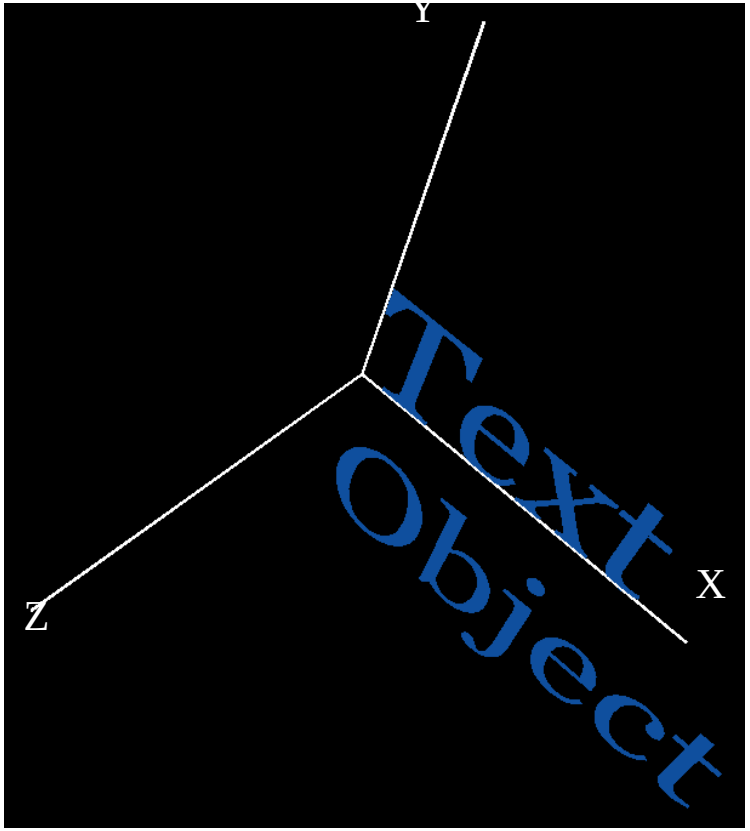


Extrusion {
  eventIn MFVec2f set_crossSection
  eventIn MFRotation set_orientation
  eventIn MFVec2f set_scale
  eventIn MFVec3f set_spine
  field SFBool **beginCap TRUE**
  field SFBool **ccw TRUE**
  field SFBool **convex TRUE**
  field SFFloat **creaseAngle 0** # [0, ∞)
  field MFVec2f **crossSection**
    [1 1, 1 -1, -1 -1, -1 1, 1 1] # (-∞ , ∞)
  field SFBool **endCap TRUE**
  field MFRotation **orientation**
    0 0 1 0 # [-1,1],(-∞ , ∞)
  field MFVec2f **scale 1 1** # (0, ∞)
  field SFBool **solid TRUE**
  field MFVec3f **spine**
    [0 0 0, 0 1 0] # (- ∞, ∞) **}**



Text { exposedField MFString **string []**
exposedField SFNode **fontStyle NULL**
exposedField MFFloat **length []** # [0, ∞)
exposedField SFFloat **maxExtent 0.0** # [0, ∞) **}**

FontStyle { field MFString **family "SERIF"**
field SFBool **horizontal TRUE**
field MFString **justify "BEGIN"**
field SFString **language ""**
field SFBool **leftToRight TRUE**
field SFFloat **size 1.0 # (**0, ∞**)**
field SFFloat **spacing 1.0 # [**0,∞ **)**
field SFString **style "PLAIN"**
field SFBool **topToBottom TRUE }**



Elementary Geometric Transformations

Transform { eventIn MFNode addChildren
eventIn MFNode removeChildren
exposedField SFVec3f **center 0 0 0** # (-∞, ∞)
exposedField MFNode **children []**
exposedField SFRotation **rotation 0 0 1 0** # [-1,1],(-∞, ∞)
exposedField SFVec3f **scale 1 1 1** # (0, ∞)
exposedField SFRotation **scaleOrientation 0 0 1 0** # [-1,1],(-∞, ∞)
exposedField SFVec3f **translation 0 0 0** # (-∞, ∞)
field SFVec3f **bboxCenter 0 0 0** # (-∞, ∞)
field SFVec3f **bboxSize -1 -1 -1** # (0, ∞) or -1,-1,-1 **}**

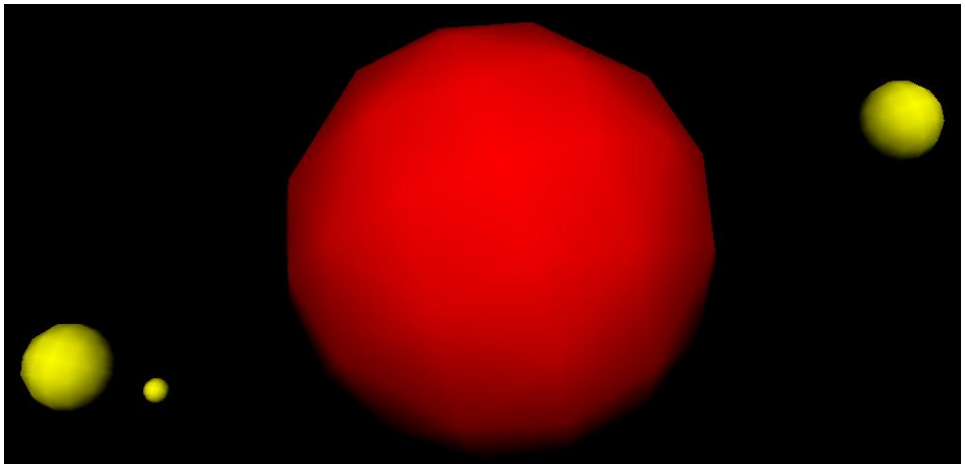
Transform { center C rotation R scale S scaleOrientation SR translation T children [...] }

Transform { translation T children
Transform { translation C children
Transform { rotation R children
Transform { rotation SR children
Transform { scale S children
Transform { rotation -SR children
Transform { translation -C children [...] }}}}}}}

(equivalent representations)

Code Examples

```
#VRML V2.0 utf8
Transform { children [
 DirectionalLight {direction 0 0 -1}
 Transform {rotation 0 0 1 0.3 scale 2 2 2 translation 0 0 3
  children [
   Shape {geometry Sphere {}
    appearance Appearance {material Material {diffuseColor 1 0 0}}}
   Transform {translation 2 0 0
    children
     DEF Ball Shape {geometry Sphere {radius .2}
      appearance Appearance {
       material Material {diffuseColor 1 1 0}}}}
   Transform {rotation 0 0 1 -0.5 translation -2 0 0
    children [
     Transform {
      children USE Ball}
     Transform {
      translation 0.3 0 -0.3
      scale 0.3 0.3 0.3
      children USE Ball}]}]}]}
```



```
#VRML V2.0 utf8

Transform {children [

 Shape {geometry IndexedFaceSet {
  color Color {color [ 1 0 0,0 1 0,0 0 1,1 1 1]}
  coord Coordinate {point [1 0 0,0 1 0,0 0 1,1 1 1]}
  coordIndex [1 0 3 -1,2 1 3 -1,0 2 3 -1,0 1 2]
  colorIndex [1 0 3 -1,2 1 3 -1,0 2 3 -1,0 1 2] colorPerVertex TRUE
  normal NULL texCoord NULL ccw FALSE convex FALSE solid TRUE
  creaseAngle 0 normalIndex [] normalPerVertex TRUE texCoordIndex []}}

 Shape {geometry ElevationGrid {
  color NULL  normal NULL  texCoord NULL
  height [0 0 0 0 0 0 0 0,0 .1 .1 .2 .4 .2 .1 0,0 .1 .1 .2 .4 .2 .1 0,
   0 0 0 0 0 0 0 0]
  ccw TRUE colorPerVertex TRUE creaseAngle 0.0 normalPerVertex TRUE
  solid FALSE xDimension 8 xSpacing 0.5 zDimension 4 zSpacing 0.5}
  appearance Appearance {material Material {diffuseColor 0 1 1}}}

 Shape {geometry Extrusion {
  beginCap TRUE ccw FALSE convex TRUE creaseAngle 0
  crossSection [1 0,.3 .3,0 1,-.3 .3 -1 0,-.3 -0.3,0 -1,.3 -.3,1 0]
  endCap TRUE solid TRUE
  orientation 0 0 1 0
  scale [1 1,0.5 0.5,0.5 0.5,1 1]
  spine [0 0 0,0 1 0,0 2 -1,0 2 -2]}
  appearance Appearance {material Material {diffuseColor 1 0 1}}} ]}
```

5. VRML: Material, Illumination, and Observer

Prof. Dr.-Ing. habil. Wolfgang Oertel

Scene design defines for Scene objects the following features:

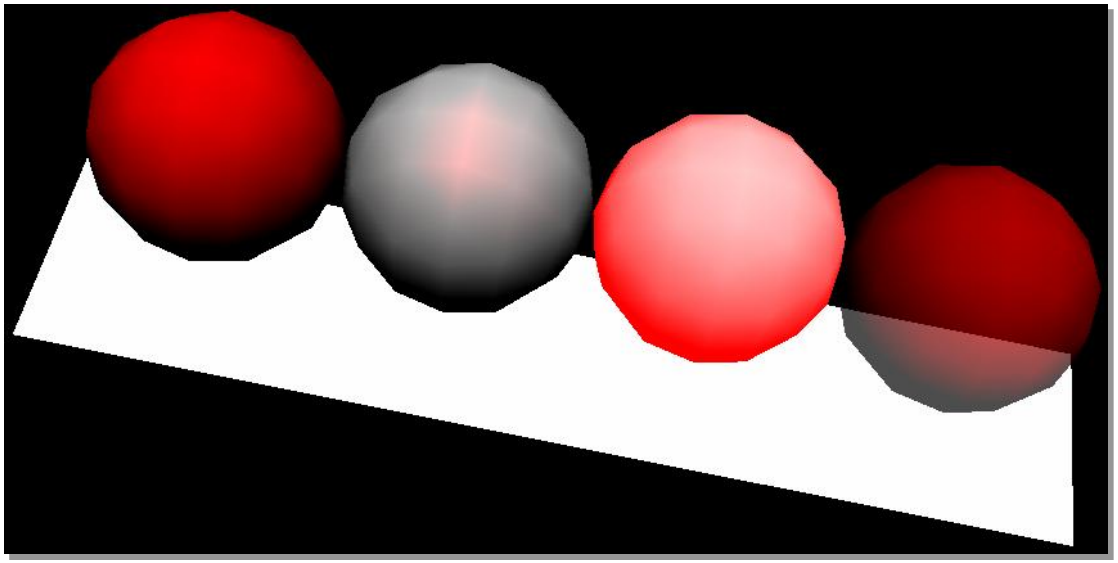
- Shape (color, material)
- Illumination (light source and propagation)
- Surface Structure (texture)
- Environment (background, view conditions)
- Optimization (details, representation)
- Observer (view point, navigation)

Set of statements determining
the appearance and the context of
objects for the observer in an efficient way

Shape

Appearance { exposedField SFNode **material NULL**
exposedField SFNode **texture NULL**
exposedField SFNode **textureTransform NULL }**

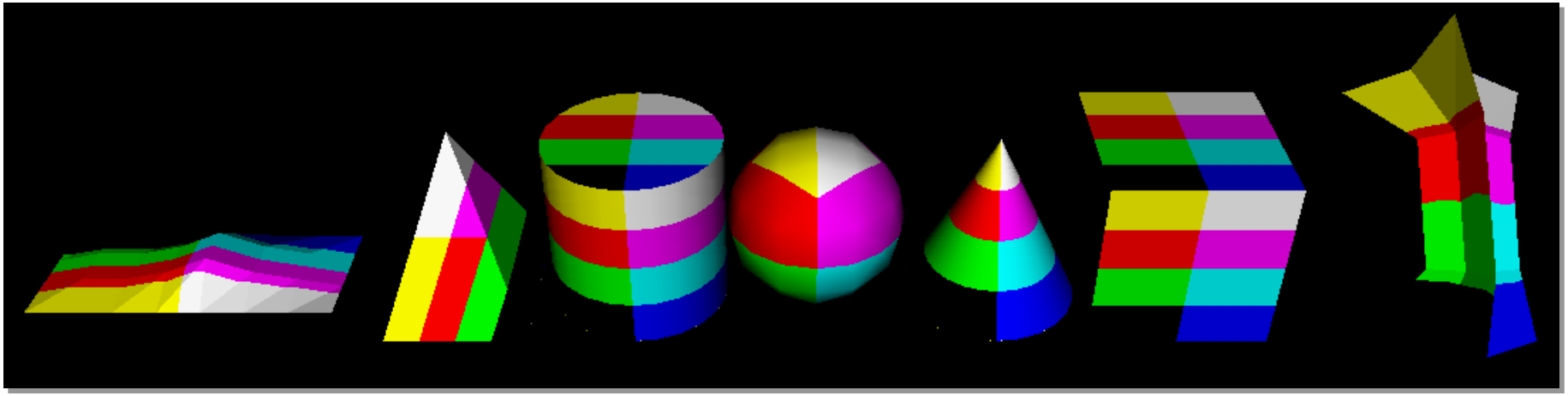
Material { exposedField SFFloat **ambientIntensity 0.2** # [0,1]
exposedField SFColor **diffuseColor 0.8 0.8 0.8** # [0,1]
exposedField SFColor **emissiveColor 0 0 0** # [0,1]
exposedField SFFloat **shininess 0.2** # [0,1]
exposedField SFColor **specularColor 0 0 0** # [0,1]
exposedField SFFloat **transparency 0** # [0,1] **}**



Surface Structure

PixelTexture { exposedField SFImage **image 0 0 0**
field SFBool **repeatS TRUE**
field SFBool **repeatT TRUE }**

TextureTransform { exposedField SFVec2f **center 0 0** # (- ∞, ∞)
exposedField SFFloat **rotation 0** # (- ∞, ∞)
exposedField SFVec2f **scale 1 1** # (- ∞, ∞)
exposedField SFVec2f **translation 0 0** # (- ∞, ∞) **}**



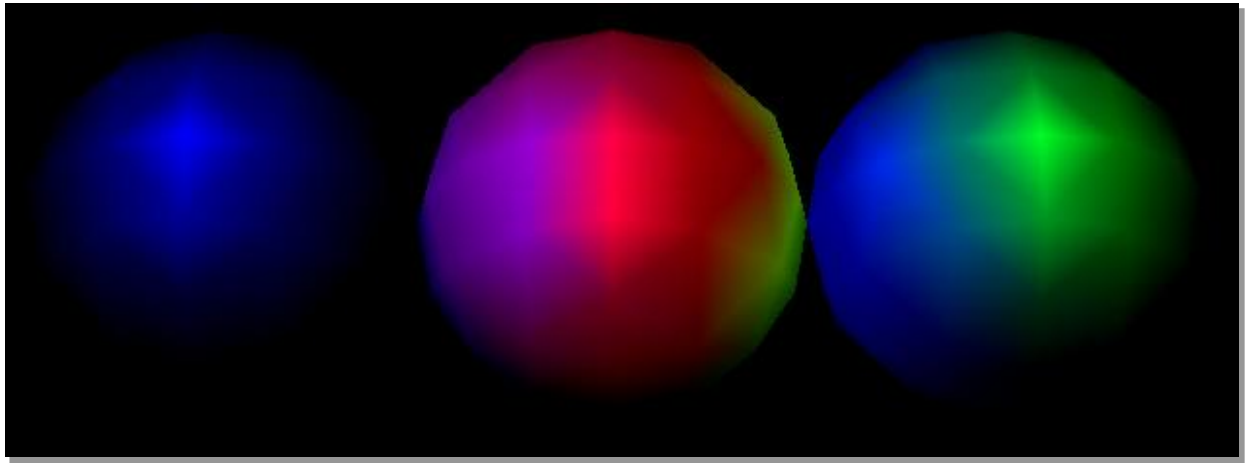
Illumination

DirectionalLight { exposedField SFFloat **ambientIntensity 0** # [0,1]
exposedField SFColor **color 1 1 1** # [0,1]
exposedField SFVec3f **direction 0 0 -1** # $(-\infty, \infty)$
exposedField SFFloat **intensity 1** # [0,1]
exposedField SFBool **on TRUE }**

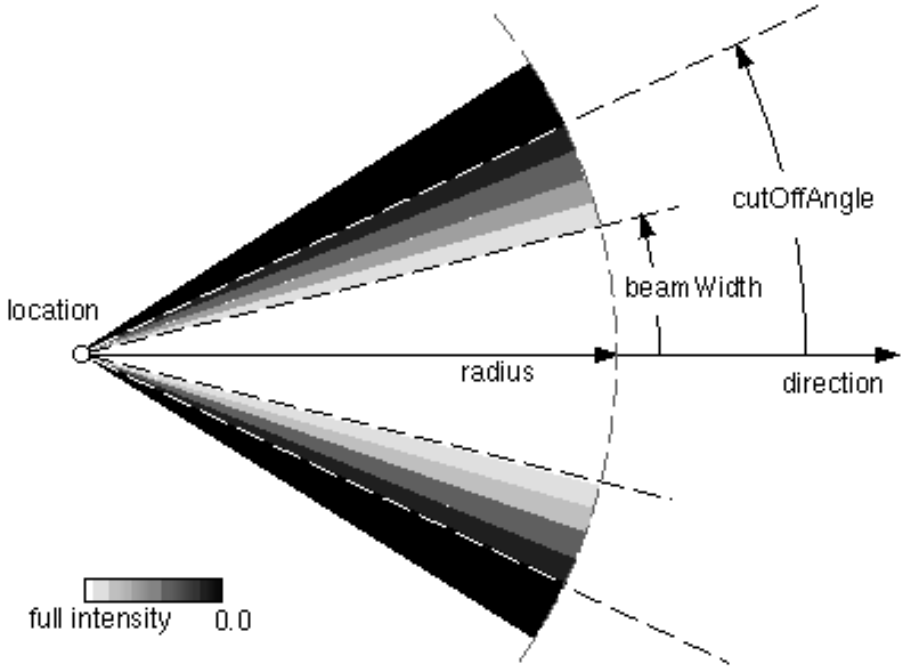
PointLight { exposedField SFFloat **ambientIntensity 0** # [0,1]
exposedField SFVec3f **attenuation 1 0 0** # $[0, \infty)$
exposedField SFColor **color 1 1 1** # [0,1]
exposedField SFFloat **intensity 1** # [0,1]
exposedField SFVec3f **location 0 0 0** # $(-\infty, \infty)$
exposedField SFBool **on TRUE**
exposedField SFFloat **radius 100** # $[0, \infty)$ **}**

Attenuation:

$$\frac{1}{\max(a[0] + a[1]r + a[2]r^2, 1)}$$



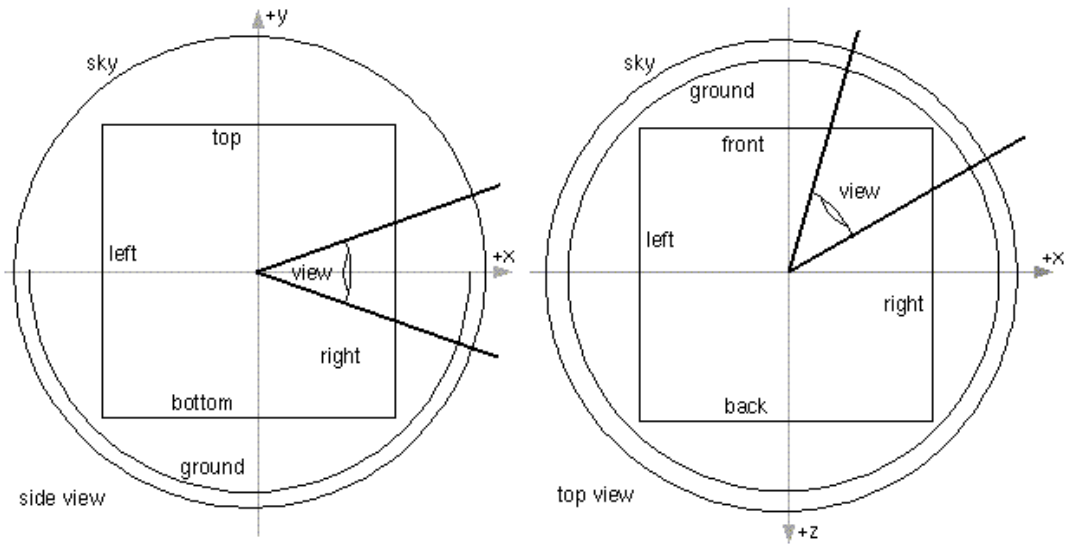
SpotLight { exposedField SFFloat **ambientIntensity 0** # [0,1]
exposedField SFVec3f **attenuation 1 0 0** # [0, ∞)
exposedField SFFloat **beamWidth 1.570796** # (0, π/2]
exposedField SFColor **color 1 1 1** # [0,1]
exposedField SFFloat **cutOffAngle 0.785398** # (0, π/2]
exposedField SFVec3f **direction 0 0 -1** # (- ∞, ∞)
exposedField SFFloat **intensity 1** # [0,1]
exposedField SFVec3f **location 0 0 0** # (- ∞, ∞)
exposedField SFBool **on TRUE**
exposedField SFFloat **radius 100** # [0, ∞) **}**



Environment

Background { eventIn SFBool set_bind
exposedField MFFloat **groundAngle []** # [0, π /2]
exposedField MFColor **groundColor []** # [0,1]
exposedField MFString **backUrl []**
exposedField MFString **bottomUrl []**
exposedField MFString **frontUrl []**
exposedField MFString **leftUrl []**
exposedField MFString **rightUrl []**
exposedField MFString **topUrl []**
exposedField MFFloat **skyAngle []** # [0, π]
exposedField MFColor **skyColor 0 0 0** # [0,1]
eventOut SFBool isBound **}**

Fog {
exposedField SFColor
color 1 1 1 # [0,1]
exposedField SFString
fogType "LINEAR"
exposedField SFFloat
visibilityRange 0 # [0, ∞)
eventIn SFBool set_bind
eventOut SFBool isBound **}**

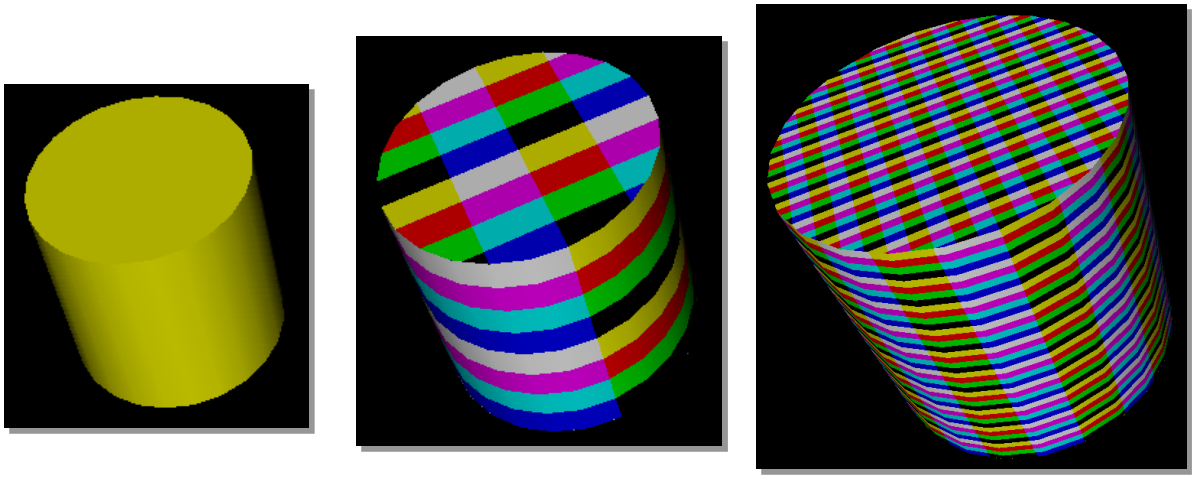


Optimization

Billboard { eventIn MFNode addChildren
eventIn MFNode removeChildren
exposedField SFVec3f **axisOfRotation 0 1 0** # (- ∞, ∞)
exposedField MFNode **children []**
field SFVec3f **bboxCenter 0 0 0** # (- ∞, ∞)
field SFVec3f **bboxSize -1 -1 -1** # (0, ∞) or -1,-1,-1 **}**

LOD { exposedField MFNode **level []**
field SFVec3f **center 0 0 0** # (- ∞, ∞)
field MFFloat **range []** # (0, ∞) **}**

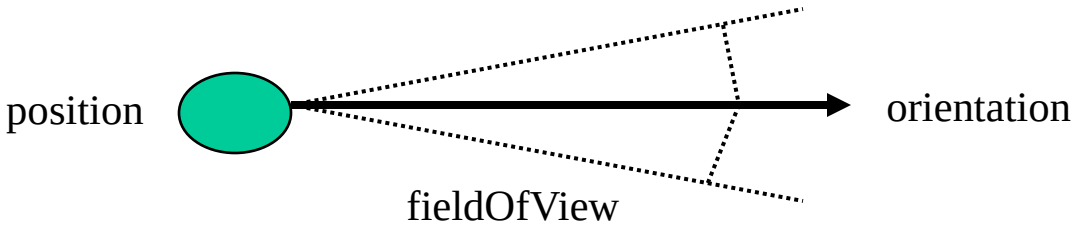
Switch { exposedField MFNode **choice []**
exposedField SFInt32 **whichChoice -1** # [-1, ∞) **}**



Observer

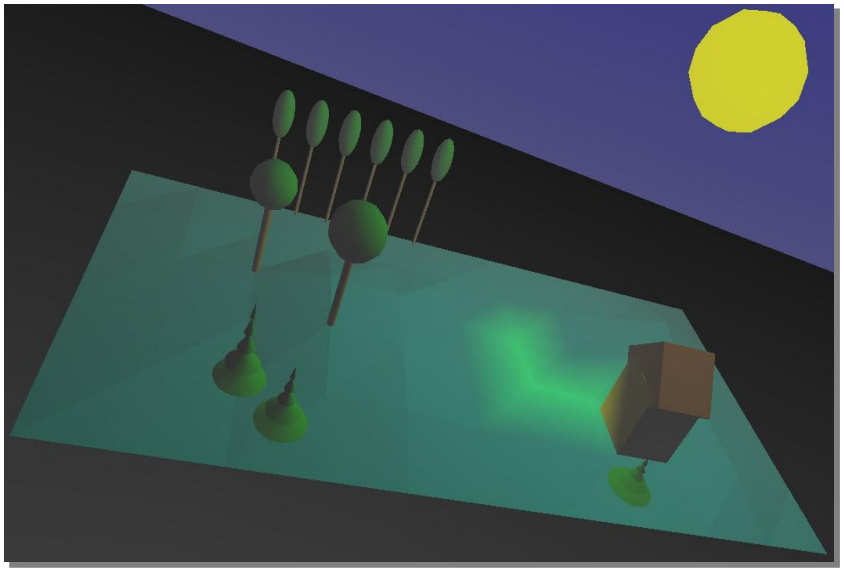
Viewpoint { eventIn SFBool set_bind
exposedField SFFloat **fieldOfView 0.785398** # (0, π)
exposedField SFBool **jump TRUE**
exposedField SFRotation **orientation 0 0 1 0** # [-1,1],(- ∞, ∞)
exposedField SFVec3f **position 0 0 10** # (- ∞, ∞)
field SFString **description ""**
eventOut SFTime bindTime
eventOut SFBool isBound **}**

NavigationInfo { eventIn SFBool set_bind
exposedField MFFloat **avatarSize [0.25, 1.6, 0.75]** # [0, ∞)
exposedField SFBool **headlight TRUE**
exposedField SFFloat **speed 1.0** # [0,∞)
exposedField MFString **type ["WALK", "ANY"]**
exposedField SFFloat **visibilityLimit 0.0** # [0, ∞)
eventOut SFBool isBound **}**



Code Example

```
#VRML V2.0 utf8
Group {children [
  NavigationInfo {headlight FALSE}
  ViewPoint {position 0 10 20 orientation 0 0 1 0.5 fieldOfView 1.0}
  SpotLight {color 1 1 0 direction 0 -1 0 radius 15}
  Fog {color 0.5 0.5 0.5 fogType "EXPONENTIAL" visibilityRange 30}
  Background {skyAngle [1.57] groundAngle [1.57]
   skyColor [0 0 .5,.3 .3 .5] groundColor [.4 .4 .4,.1 .1 .1]}

Transform {translation -5 0 0 children [
  PointLight {intensity 0.5 location 0 0 0 radius 12}
  Shape {geometry Sphere {radius 0.6}
   appearance Appearance {material Material {emissiveColor 1 1 0}}}]}

Transform {translation -5 0 0
 children [
  Shape {
   geometry ElevationGrid {
    color NULL normal NULL
    height [0 0 0 0 0 0 0 0 0 0 0,
     0 .1 .1 .1 .1 .1 0 0 .1 .3 0,...]
    colorPerVertex TRUE
    normalPerVertex TRUE
    xDimension 11 xSpacing 1
    zDimension 6 zSpacing 1}
   appearance Appearance {
    material Material {diffuseColor 0 1 0.8 }}}]}
```



6. VRML: Animation and Interaction

Prof. Dr.-Ing. habil. Wolfgang Oertel

Animation and **Interaction** means:

- Computation of motion and changing sequences for objects
- Logical decisions in the case of alternatives
- Reaction to internal events
- Reaction to external events
- Changeable nodes as premise
- Elementary or complex processes

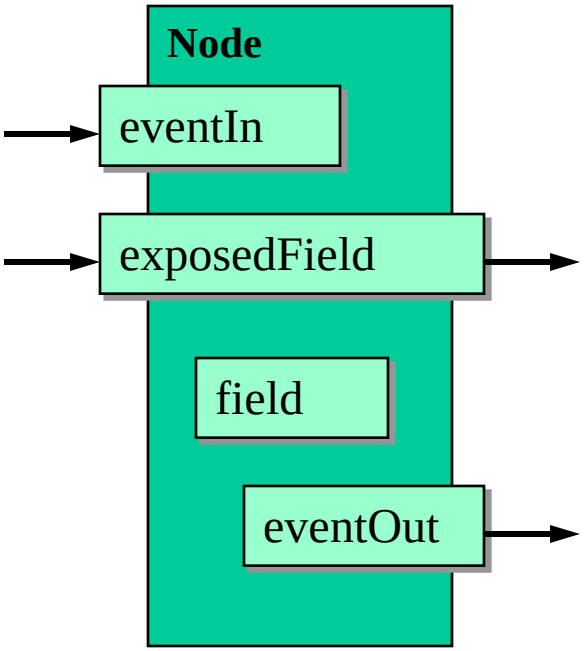
Realization by:

- Interpolators
- Sensors
- Routes
- Navigations
- Scripts

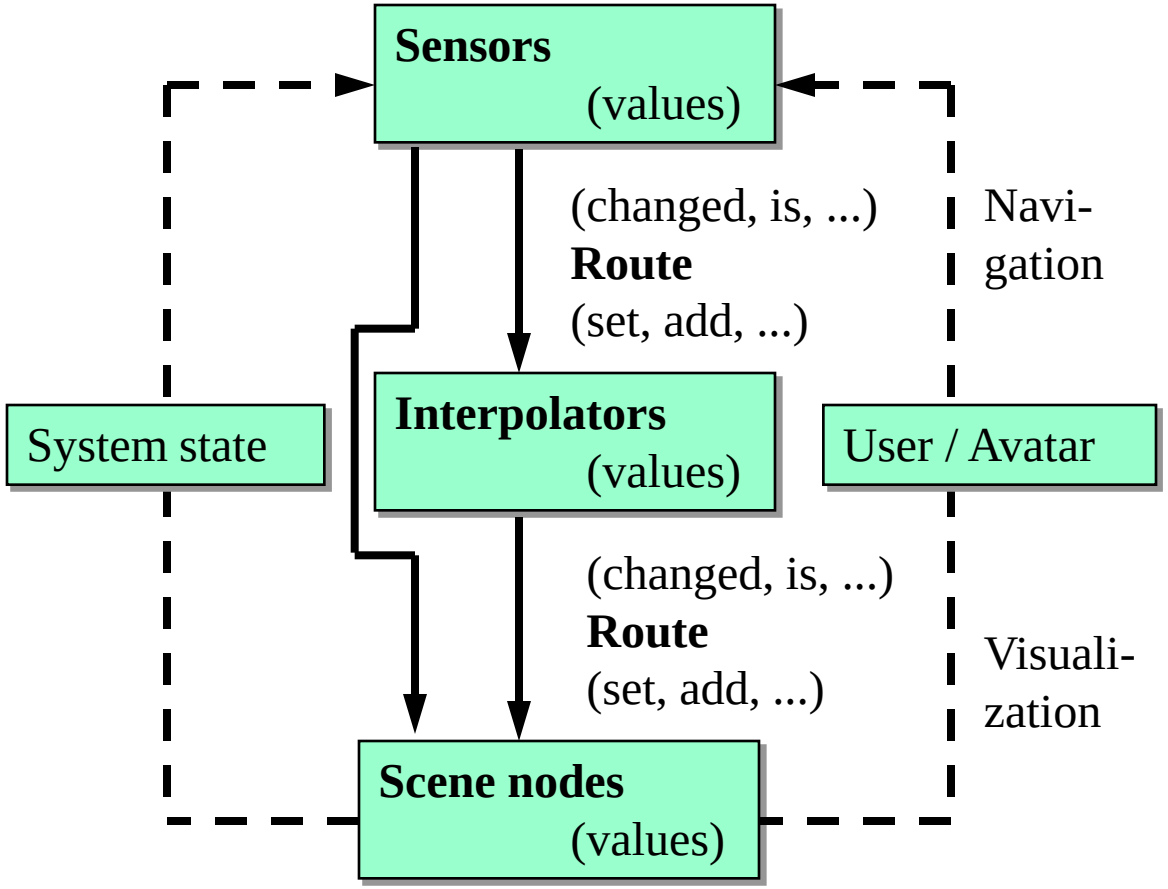
Result: changing and changed world

Working Principle

Event processing:



Event propagation:



Principle: Processes change states

Time and Touch Sensors

```
TimeSensor { exposedField SFTime cycleInterval 1 # (0, ∞)
             exposedField SFBool enabled TRUE
             exposedField SFBool loop FALSE
             exposedField SFTime startTime 0 # (-∞ , ∞ )
             exposedField SFTime stopTime 0 # (- ∞, ∞)
             eventOut SFTime cycleTime
             eventOut SFFloat fraction_changed # [0, 1]
             eventOut SFBool isActive
             eventOut SFTime time }
```

```
TouchSensor { exposedField SFBool enabled TRUE
              eventOut SFVec3f hitNormal_changed
              eventOut SFVec3f hitPoint_changed
              eventOut SFVec2f hitTexCoord_changed
              eventOut SFBool isActive
              eventOut SFBool isOver
              eventOut SFTime touchTime }
```

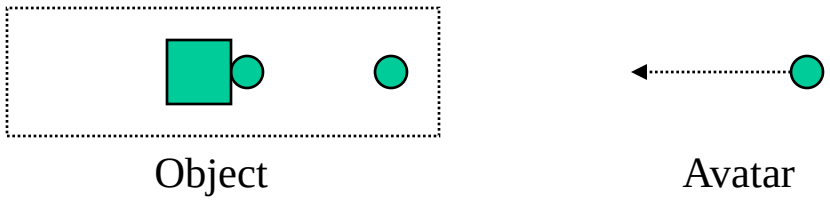
Other interactive node: **Anchor**

Navigation Sensors

```
VisibilitySensor { exposedField SFVec3f center 0 0 0 # (- ∞, ∞)
                   exposedField SFBool enabled TRUE
                   exposedField SFVec3f size 0 0 0 # [0, ∞)
                   eventOut SFTime enterTime
                   eventOut SFTime exitTime
                   eventOut SFBool isActive }
```

```
ProximitySensor { exposedField SFVec3f center 0 0 0 # (- ∞, ∞)
                  exposedField SFVec3f size 0 0 0 # [0, ∞)
                  exposedField SFBool enabled TRUE
                  eventOut SFBool isActive
                  eventOut SFVec3f position_changed
                  eventOut SFRotation orientation_changed
                  eventOut SFTime enterTime
                  eventOut SFTime exitTime }
```

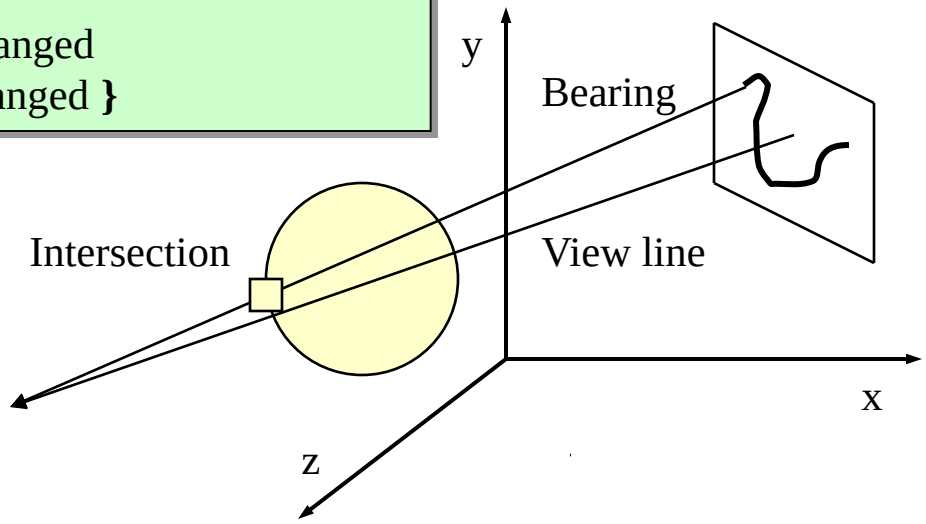
Other navigation nodes: **Collision, LOD**



Pulling Sensors

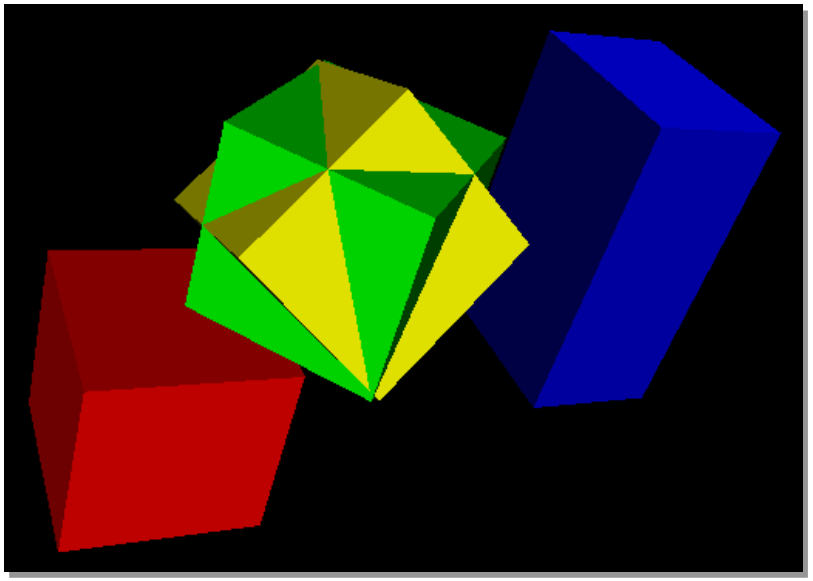
```
PlaneSensor { exposedField SFBool autoOffset TRUE
              exposedField SFBool enabled TRUE
              exposedField SFVec2f maxPosition -1 -1 # (- ∞, ∞)
              exposedField SFVec2f minPosition 0 0 # (- ∞, ∞)
              exposedField SFVec3f offset 0 0 0 # (- ∞ , ∞ )
              eventOut SFBool isActive
              eventOut SFVec3f trackPoint_changed
              eventOut SFVec3f translation_changed }
```

```
SphereSensor { exposedField SFBool autoOffset TRUE
               exposedField SFBool enabled TRUE
               exposedField SFRotation offset 0 1 0 0 # [-1,1],(-∞ ,∞ )
               eventOut SFBool isActive
               eventOut SFRotation rotation_changed
               eventOut SFVec3f trackPoint_changed }
```



CylinderSensor { exposedField SFBool **autoOffset TRUE**
exposedField SFFloat **diskAngle 0.262** # (0, π/2)
exposedField SFBool **enabled TRUE**
exposedField SFFloat **maxAngle -1** # [-2π ,2π]
exposedField SFFloat **minAngle 0** # [-2π ,2 π]
exposedField SFFloat **offset 0** # (- ∞ , ∞)
eventOut SFBool isActive
eventOut SFRotation rotation_changed
eventOut SFVec3f trackPoint_changed **}**

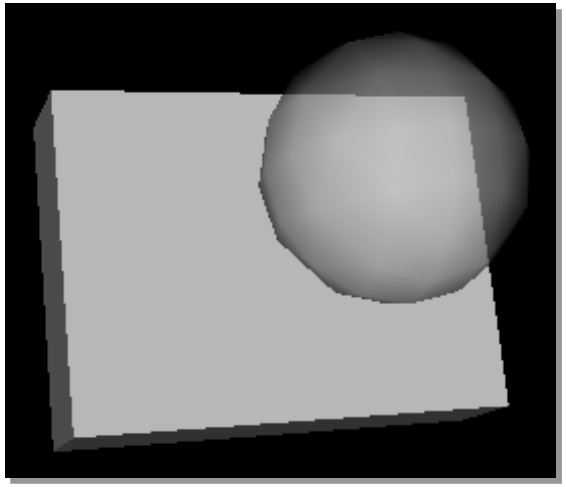
```
#VRML V2.0 utf8
Group {children [
  DEF Translator PlaneSensor {}
  DEF Cube1 Transform {
    children Shape {geometry Box {}}}]}
Group {children [
  DEF RotatorS SphereSensor {}
  DEF Cube3 Transform {
    children Shape {geometry Box {}}}]}
ROUTE Translator.translation_changed
  TO Cube1.set_translation
ROUTE RotatorS.rotation_changed
  TO Cube3.set_rotation   ...
```



Scalar and Color Interpolators

ScalarInterpolator { eventIn SFFloat set_fraction # (- ∞, ∞)
exposedField MFFloat **key []** # (- ∞ , ∞)
exposedField MFFloat **keyValue []** # (- ∞, ∞)
eventOut SFFloat value_changed **}**

ColorInterpolator { eventIn SFFloat set_fraction # (- ∞ , ∞)
exposedField MFFloat **key []** # (- ∞, ∞)
exposedField MFColor **keyValue []** # [0,1]
eventOut SFColor value_changed **}**



```
w#VRML V2.0 utf8
Group {
  children [ ...
    DEF Ball Transform {
        children Shape {
          appearance Appearance {
            material DEF Mat Material {
              transparency 0}}
          geometry Sphere {}}}
    DEF Timer TimeSensor {cycleInterval 2 loop TRUE}
    DEF Valuesetter ScalarInterpolator {
        key [0,0.5,1.0] keyValue [0,0.5,1.0]}]}
ROUTE Timer.fraction_changed TO Valuesetter.set_fraction
ROUTE Valuesetter.value_changed TO Mat.set_transparency
```

Geometric Interpolators

CoordinateInterpolator { eventIn SFFloat set_fraction # (- ∞, ∞)
exposedField MFFloat **key []** # (-∞ ,∞)
exposedField MFVec3f **keyValue []** # (- ∞, ∞)
eventOut MFVec3f value_changed **}**

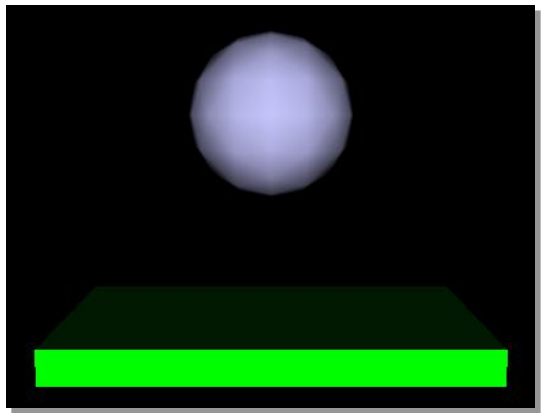
PositionInterpolator { eventIn SFFloat set_fraction # (- ∞, ∞)
exposedField MFFloat **key []** # (- ∞ , ∞)
exposedField MFVec3f **keyValue []** # (- ∞, ∞)
eventOut SFVec3f value_changed **}**

OrientationInterpolator { eventIn SFFloat set_fraction # (- ∞ , ∞)
exposedField MFFloat **key []** # (- ∞ , ∞)
exposedField MFRotation **keyValue []** # [-1,1],(- ∞ , ∞)
eventOut SFRotation value_changed **}**

NormalInterpolator { eventIn SFFloat set_fraction # (-∞ ,∞)
exposedField MFFloat **key []** # (-∞, ∞)
exposedField MFVec3f **keyValue []** # (- ∞, ∞)
eventOut MFVec3f value_changed **}**

Code Example

```
#VRML V2.0 utf8
Transform {rotation 1 0 0 0.1, translation 0 -1 0,
  children [
    Transform {translation 0 -0.8 0
      children [Shape {geometry Box {size 5 0.4 3}
         appearance Appearance {
          material Material {diffuseColor 0 1 0}}},
        DEF Switch TouchSensor {}]}
    DEF Motion Transform {
      children [Shape {geometry Sphere {radius 1.0}
          appearance Appearance {
               material DEF Color Material {}}}]}
    DEF Light DirectionalLight {direction 0 -10 -10, on FALSE},
    DEF Chronos TimeSensor {cycleInterval 4 loop TRUE},
    DEF PosCalc PositionInterpolator {
      key [0.0,0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1.0,1.1]
      keyValue [0 0.0 0,0 1.5 0,0 2.3 0,0 2.7 0,0 2.9 0,0 3.0 0,
                0 2.9 0,0 2.7 0,0 2.3 0,0 1.5 0, 0 0.0 0]},
    DEF ColCalc ColorInterpolator {
      key [0.0,0.3,0.5,0.6,0.7,0.9,1.1]
      keyValue [1 1 1,1 1 1,0 1 0,1 0 0,0 0 1,1 1 1,1 1 1]} ]}
ROUTE Switch.isActive TO Licht.on
ROUTE Chronos.fraction_changed TO PosCalc.set_fraction
ROUTE PosCalc.value_changed TO Motion.set_translation
ROUTE Chronos.fraction_changed TO ColCalc.set_fraction
ROUTE ColCalc.value_changed TO Color.set_diffuseColor
```



7. VRML: Programming and Networking

Prof. Dr.-Ing. habil. Wolfgang Oertel

World design by connecting internal and external objects and scenes:

- Definition and use of nodes
- Definition and use of prototypes
- Links to other documents
- Use of external languages

→ Hierarchy or network of objects
distributed over several files or computers

→ Functionality of a universal programming language

Definitions and Prototypes

DEF:	Naming of an existing node
USE:	Reference to a named node
PROTO:	File-internal declaration of a new node type
EXTERNPROTO:	File-external declaration of a new node type
IS:	Mapping of fields and events of the interface

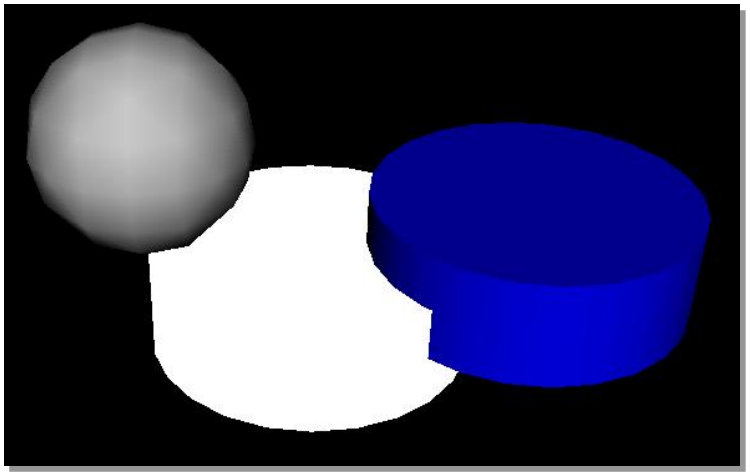
Annotations:
- Reuse of a defined node (DEF) by a reference (USE)
- No change or copy of the original node, but embedding it in the actual context (Color, Transformation, …)
- Declaration and definition of prototypes by PROTO and EXTERNPROTO
- Use of IS within prototype definitions
- prototype argument types: exposedField, field, eventIn, eventOut

Prototype declaration

Prototype definition	**exposedField**	**field**	**eventIn**	**eventOut**
exposedField	yes	yes	yes	yes
field	no	yes	no	no
eventIn	no	no	yes	no
eventOut	no	no	no	yes

Examples:

```
DEF Cyl1 Cylinder {height 2 radius 3}
Transform {translation 4 1 0
  children [Shape {geometry USE Cyl1
    appearance Appearance {
      material Material {
        diffuseColor 0 0 1}}}]}
```



```
PROTO GeometryObject [
  exposedField SFVec3f trans 0 0 0
  exposedField SFColor color 0.8 0.8 0.8
  exposedField SFNode geom NULL] {
  Transform {translation IS trans
    children [Shape {geometry IS geom
      appearance Appearance {
        material Material {
          diffuseColor IS color}}}]}}
GeometryObject {
  trans -3 3 0 geom Sphere {radius 2}}
```

File1:

```
EXTERNPROTO Glas []
  ["http://...material.wrl#Glas"]
appearance Appearance {material Glas{}}
```

File2:

```
PROTO Glas [] {Material {...}}
PROTO Metal [] {Material {...}}
```

References

```
Inline { exposedField MFString url []
         field SFVec3f bboxCenter 0 0 0 # (- ∞, ∞)
         field SFVec3f bboxSize -1 -1 -1 # (0, ∞) or -1,-1,-1 }
```

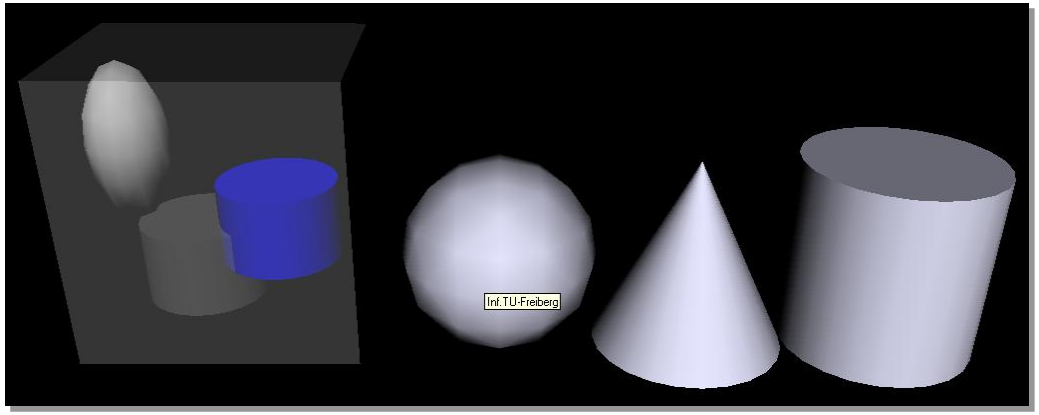
```
Anchor { eventIn MFNode addChildren
         eventIn MFNode removeChildren
         exposedField MFNode children []
         exposedField SFString description ""
         exposedField MFString parameter []
         exposedField MFString url []
         field SFVec3f bboxCenter 0 0 0 # (- ∞, ∞)
         field SFVec3f bboxSize -1 -1 -1 # (0, ∞) or -1,-1,-1 }
```

Annotations:

- Insert the contents of an arbitrary VRML file from the web by Inline
- Definition of a hyperlink to an arbitrary document in the web by Anchor (VRML, HTML, XML)
- Definition of observer coordinates by Viewpoint
- Use of the HTML frame concept
- Execution of operations at runtime (if necessary)

Examples:

```
Transform {translation -3 0 0 scale 0.2 0.4 0.2
  children [Inline {url "Example7a.wrl"}]}

Anchor {url "http://www.informatik,htw-dresden.de"
  description "Informatik HTW Dresden"
  children [Shape {geometry Sphere {}}]}

Anchor {url "#View2" description "View2"
  children [Transform {translation 2 0 0
    children [ Shape {geometry Cone {}}]}]}

Anchor {url "Example7a.wrl"
  children [Transform {translation 4 0 0
    children [ Shape {geometry Cylinder {}}]}]}

DEF View1 Viewpoint {description "View1"}

DEF View2 Viewpoint {position 0 10 0
  orientation 1 0 0 -1.5 description "View2"}
```



Link from HTML to VRML:

```
<A HREF="Example7b.wrl">
  Virtual World </A>
```

Scripts

```
Script { exposedField MFString url []
         field SFBool directOutput FALSE
         field SFBool mustEvaluate FALSE
         # And any number of:
         eventIn eventType eventName
         field fieldType fieldName initialValue
         eventOut eventType eventName }
```

Annotations:
- Task: Description of procedural sequences and logical decisions
- Realization: Sequence of interpretable commands of an external language
- Localization: Function in the source code or in a separate file
- Form: Fields for storage of values, events for transfer of values
  one function with the same name for each eventIn field
- Persistence: for script node fields, not for local variables
- Event processing: Receiving of an event → activation of the respective function → providing the results (by ROUTE)
- Node access: to all fields of the own node and viewable fields of other nodes (by name or USE)
- Browser communication: information and activation
- Special functions: pre- and post-processing

Primary language: JavaScript
other languages system dependent (Java, C/C++, VB, TCL)

JavaScript

Interface:

Function design:
```
url "javascript: function Name (wert,zeit) {…}"
url "demo//program1.js"
```

Event processing:
```
eventname, value, time
```

Node access:
```
node.name=expression; variable=node.name_changed;
node.name=node.name
```

Data types:
```
SFBOOL → boolean; SFFloat →float; SFInt32 →int;
SFString → string;
SFVec3f, MFFloat, … → array[], array[][], …
```

Math-, Date-, String-Objekt: `Math.methode(…), Date.methode(…), text.methode(…)`

Browser-Object: `browser.getName(), browser.getVersion(), getCurrentSpeed(), getCurrentFrameRate(), getWorldURL(), replaceWorld(nodes), loadURL(url,parameter), setDescription(description), createVrmlFromString(vrmlSyntax), createVrmlFromURL(url,node,event), addRoute(fromNode,fromEventOut,toNode,toEventIn), deleteRoute(fromNode,fromEventOut,toNode,toEventIn)`

Language elements:

Comment: `/* ... */, //`

Function: `function name (parameterlist) {statements}`

Variable: `var name1, name2, ...;`
(one type for numbers, letters, logical values or strings

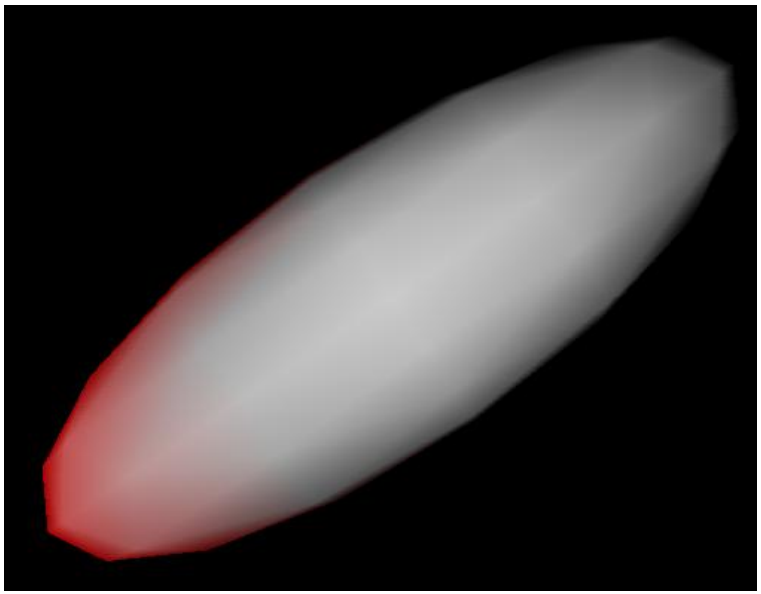
Expression: Value assignment: `=`
Arithmetic: `+, -, *, /, %, ++, --`
Logic: `&&, ||, !`
Comparison: `<, <=, ==, !=, >=, >`
String: `+, +=`

Statement: Command: `expression;`
Block: `{}`
Alternative: `if (expression) statement else statement`
`switch(expression){Case 1:statement ... default:statement}`
Loop: `for (expression; expression; expression) statement`
`do statement while (expression)`

Object: `for example: window.document.write("Text");`
(Object hierarchy with attributes and methods)

Examples:

```
DEF Ball Transform {
 children [
  Shape {geometry Sphere {}
   appearance Appearance {
    material Material {}}}
  DEF Touch TouchSensor
   {}]}

DEF Light DirectionalLight {
 color 1 0 0}
```



```
DEF Extent Script {
 eventIn SFBool Touch
 field SFNode Node USE Ball
 field SFVec3f Scal 1 1 1
 directOutput TRUE
 url "javascript:
  function Touch (value,time) {
   if (Node.scale[0] > 5) {
    Node.scale=Scal;
    Node.translation[2]-=3;}
   else {Node.scale[0]+=0.3;
    Node.rotation[3]+=0.1;}}"}

DEF Illuminate Script {
 eventIn SFRotation Rotat
 eventOut SFVec3f Direct
 url "javascript:
  function Rotat (value,time) {
    Direct[0]=Math.sin(value[3]);
    Direct[2]=Math.cos(value[3]);}"}

ROUTE Touch.isOver TO
 Extent.Touch
ROUTE Ball.rotation TO
 Illuminate.Rotat
ROUTE Illuminate.Direct TO
 Light.direction
```

Adding and Removing

Group nodes have children (set of subordinated objects)
addChildren: Adding of new children
removeChildren: Removing of existing children

Example:

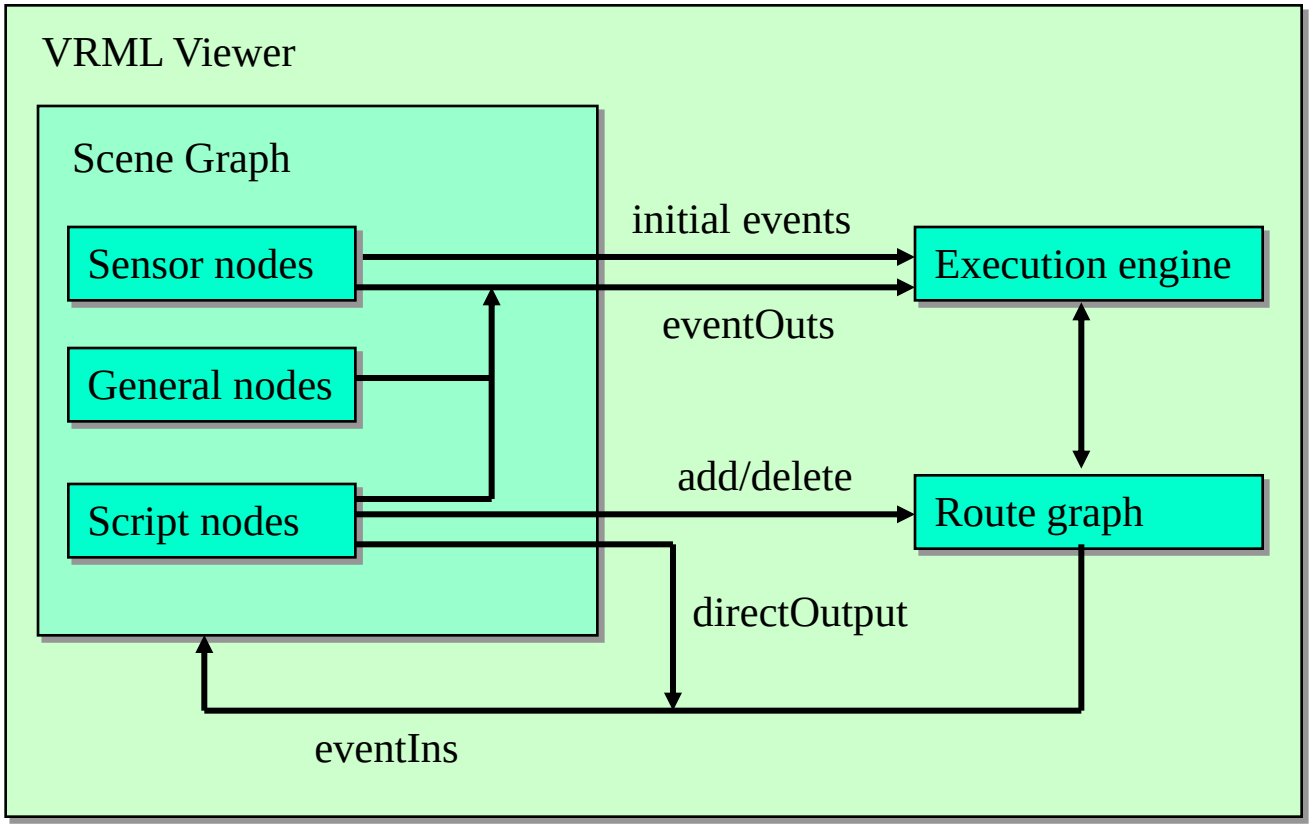
Removing of objects

```
DEF SPHERES Group {children [
 Shape{geometry Sphere{}} Shape{geometry Cone{}}
 Shape{geometry Cylinder{}}]}
Transform {children [
 Shape {geometry Box{}}
 DEF BOX_SENSOR TouchSensor{}]}
DEF REMOVER Script {
 eventIn SFBool remove_it
 eventOut MFNode new_node
 field SFNode spheres USE SPHERES
 url ["javascript: function remove_it(value) {
  if (value) {
   if (spheres.children_changed.length > 0) {
    new_node = new MFNode(spheres.children_changed[0]);}}}"]}
ROUTE BOX_SENSOR.isActive TO REMOVER.remove_it
ROUTE REMOVER.new_node TO SPHERES.removeChildren
```

Example: Adding of objects

```
DEF SPHERES Group {}
DEF HIDDEN Transform {children [Shape {geometry Sphere {}}]}
Transform {children [Shape {geometry Box{}}
 DEF BOX_SENSOR TouchSensor{}]}
DEF ADDER Script {
 eventIn SFBool add_it
 eventOut MFNode new_sphere
 field SFNode spheres USE SPHERES
 field SFNode hidden USE HIDDEN
 url ["javascript: function add_it(value) {
  if (value) {
   if (spheres.children_changed.length == 0) {
    new_sphere = Browser.createVrmlFromString(
     'Transform { translation -2 0 0 children [ ' +
     'Shape { geometry Sphere { } } ] }' );
   } else if (spheres.children_changed.length == 1) {
    new_sphere = new MFNode ( new SFNode(
     'Transform { translation 0 0 0 children [ ' +
     'Shape { geometry Sphere { } } ] }'));
   } else if (spheres.children_changed.length == 2) {
    new_sphere = new MFNode ( hidden.choice_changed[0]);
    new_sphere[0].set_translation = new SFVec3f(2,0,0); }}}"]}]
ROUTE BOX_SENSOR.isActive TO ADDER.add_it
ROUTE ADDER.new_sphere TO SPHERES.addChildren
```

Conceptual Execution Model



8. VRML: Involvement of Multimedia Documents

Prof. Dr.-Ing. habil. Wolfgang Oertel

Word design by connection with external multimedia documents:

- Images
- Videos
- Audios
- Texts
- Programs

Texts and Programs:

* by node **Anchor** as external documents
* by special nodes **Text** or **Script**

Connection of 3D modeling
with multimedia modeling

Images

ImageTexture { exposedField MFString **url []**
field SFBool **repeatS TRUE**
field SFBool **repeatT TRUE }**

```
Transform {translation 9.95 2 -3
  children [
    Shape {geometry Box {size 0.01 2.5 3}
      appearance Appearance {
        texture ImageTexture {
          url "Image1.jpg"}
        textureTransform TextureTransform {scale 1 1}}}]}
```

Context: in the same way as node **PixelTexture**
Other node: **Background**

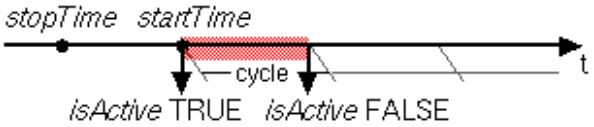
Videos

MovieTexture { exposedField SFBool **loop FALSE**
exposedField SFFloat **speed 1.0** # (-∞ ,∞)
exposedField SFTime **startTime 0** # (-∞ ,∞)
exposedField SFTime **stopTime 0** # (- ∞, ∞)
exposedField MFString **url []**
field SFBool **repeatS TRUE**
field SFBool **repeatT TRUE**
eventOut SFTime duration_changed
eventOut SFBool isActive **}**

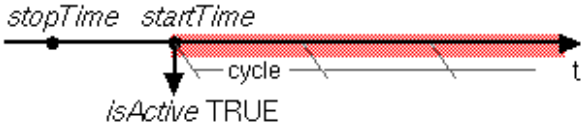
```
Transform {translation 0 2.5 -7.9
  children [DEF S3 TouchSensor {}
    Shape {geometry Box {size 6 4 0.01}
      appearance Appearance {
        texture DEF V3 MovieTexture {
          loop TRUE
          url "Video2.mpeg"
          stopTime 1 startTime 0}}}]}
ROUTE S3.touchTime TO V3.startTime
```

Play-back speed changeable

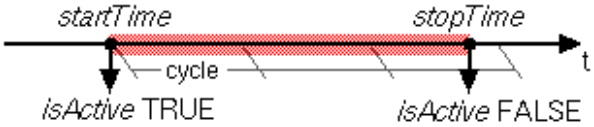
startTime >= ***stopTime, loop*** FALSE:



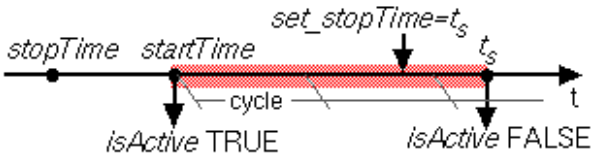
startTime >= ***stopTime, loop*** TRUE:



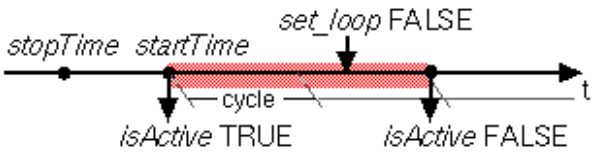
startTime < ***stopTime, loop*** TRUE:



set_stopTime, loop TRUE:



loop TRUE, ***set_loop*** FALSE:

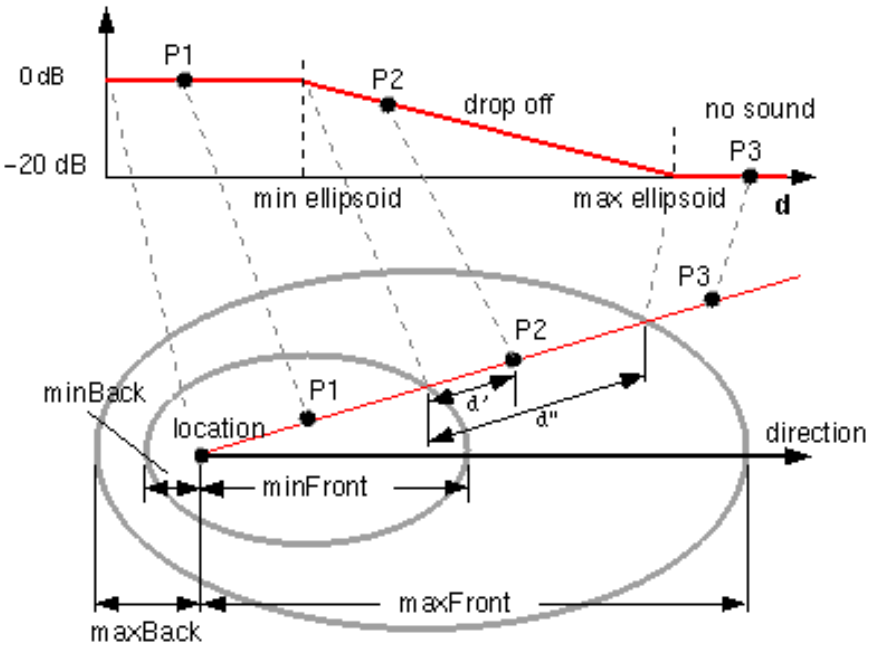


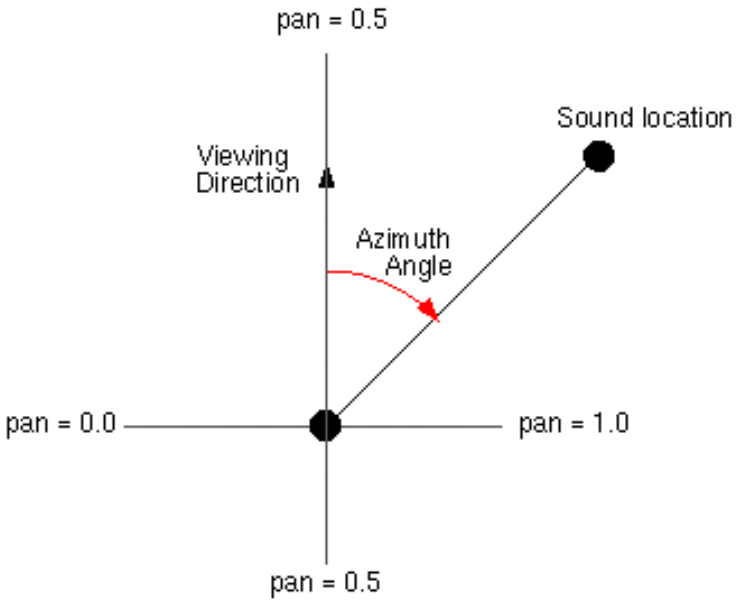
Sounds

Sound { exposedField SFVec3f **direction 0 0 1** # (- ∞, ∞)
exposedField SFFloat **intensity 1** # [0,1]
exposedField SFVec3f **location 0 0 0** # (- ∞, ∞)
exposedField SFFloat **maxBack 10** # [0, ∞)
exposedField SFFloat **maxFront 10** # [0, ∞)
exposedField SFFloat **minBack 1** # [0,∞)
exposedField SFFloat **minFront 1** # [0, ∞)
exposedField SFFloat **priority 0** # [0,1]
exposedField SFNode **source NULL**
field SFBool **spatialize TRUE }**

attenuation = -20 * (d' / d")

leftPanFactor = 1 - pan^2
rightPanFactor = 1 - (1 - pan)^2





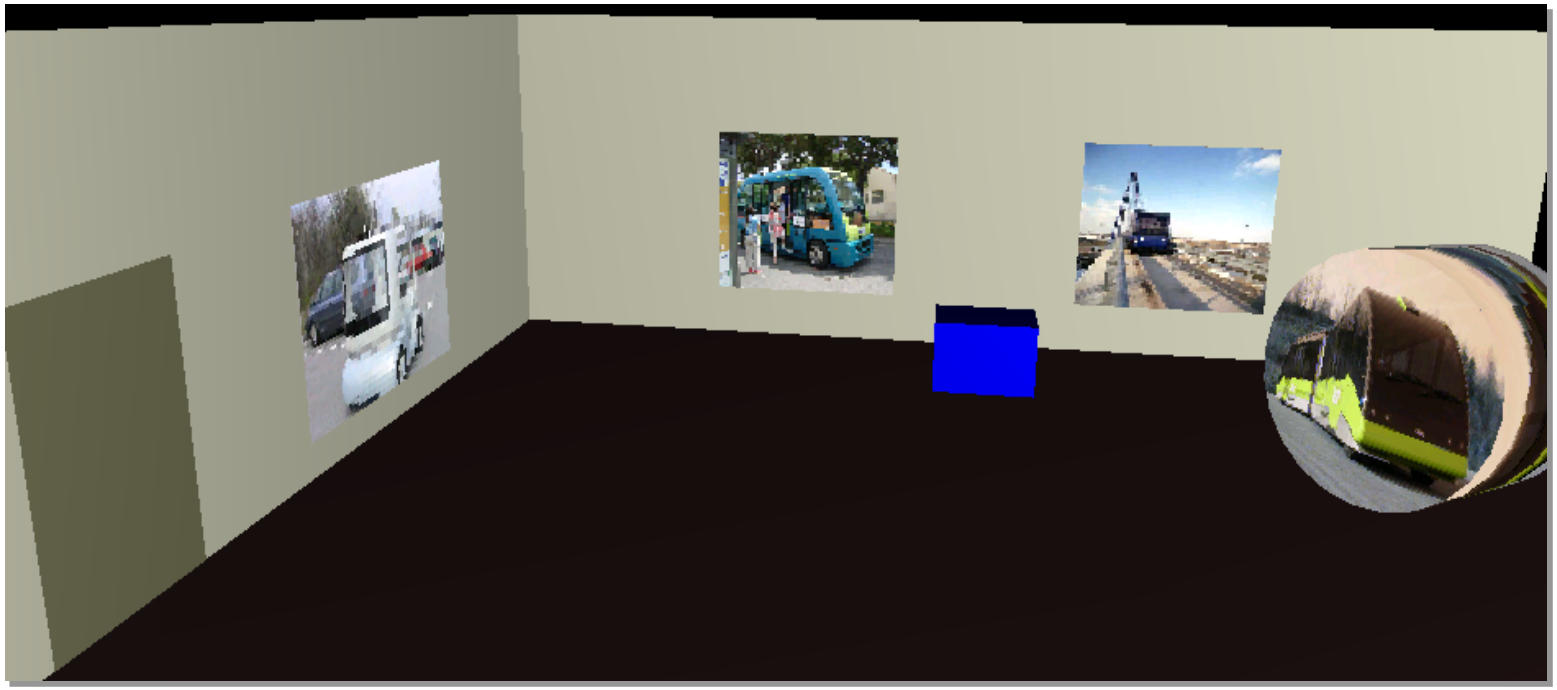
Audios

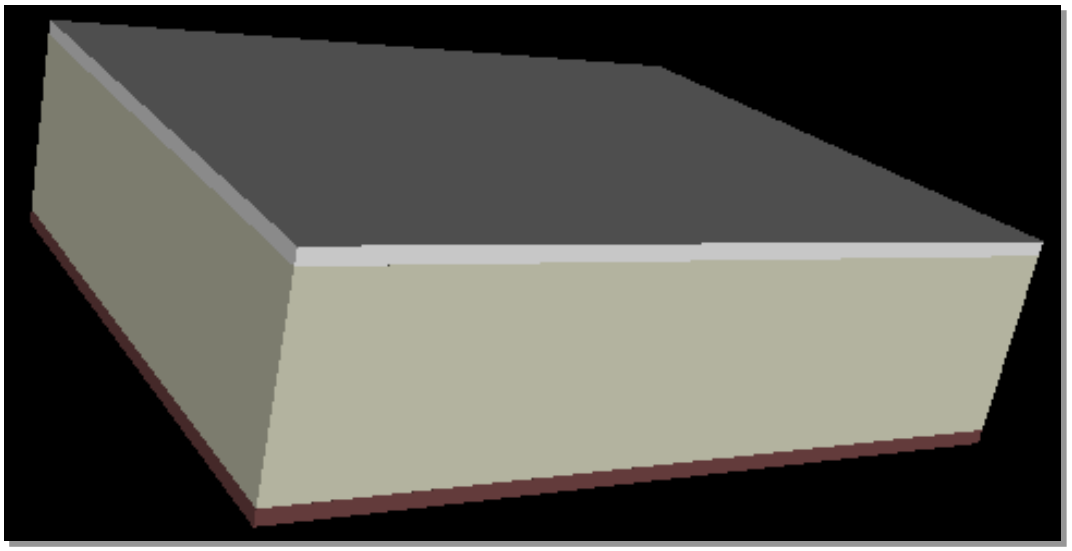
AudioClip { exposedField SFString **description** ""
exposedField SFBool **loop FALSE**
exposedField SFFloat **pitch 1.0** # (0, ∞)
exposedField SFTime **startTime 0** # (- ∞, ∞)
exposedField SFTime **stopTime 0** # (- ∞, ∞)
exposedField MFString **url []**
eventOut SFTime duration_changed
eventOut SFBool isActive **}**

```
Transform {translation -8 0.5 0
  children [
    Shape {geometry Box {size 1 1 1.5}
      appearance Appearance {
        material Material {diffuseColor 0 0 1}}}]}
Sound {location -8 1 0
  source AudioClip {loop TRUE
    url "Audio1.RMI"}}
```

Mapping between audio sources and audio channels
Play-back speed changeable in connection with pitch
Same time behavior as videos

Multimedia Room



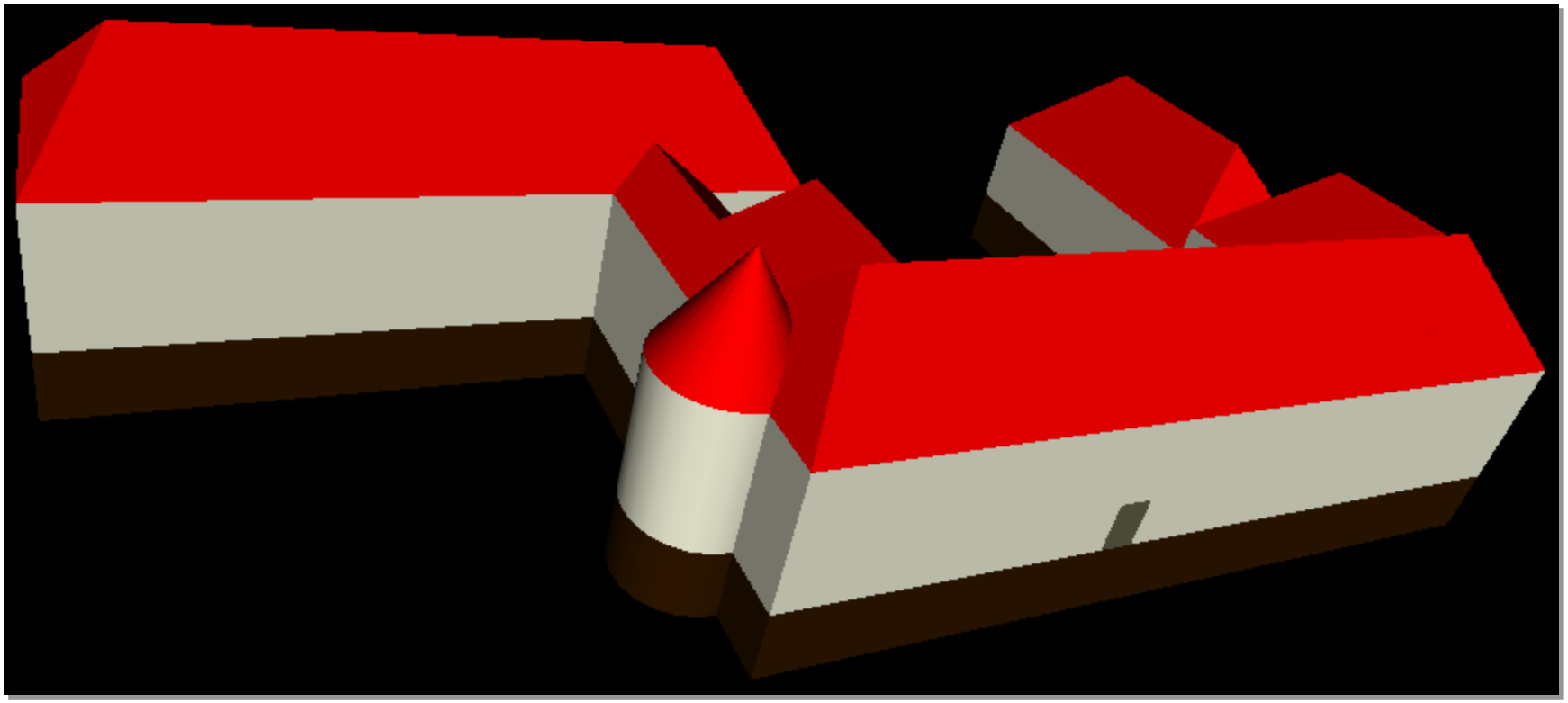


Interior and exterior view of a room

World Information

WorldInfo { field MFString **info []**
field SFString **title "" }**

```
WorldInfo {
  info ["Author: Wolfgang Oertel" "Institution: HTW Dresden"]
  title „Building"}
```



9. VRML: Intelligent Behaviour

Prof. Dr.-Ing. habil. Wolfgang Oertel

Intelligent Behaviour by integration of artificial intelligence technologies:

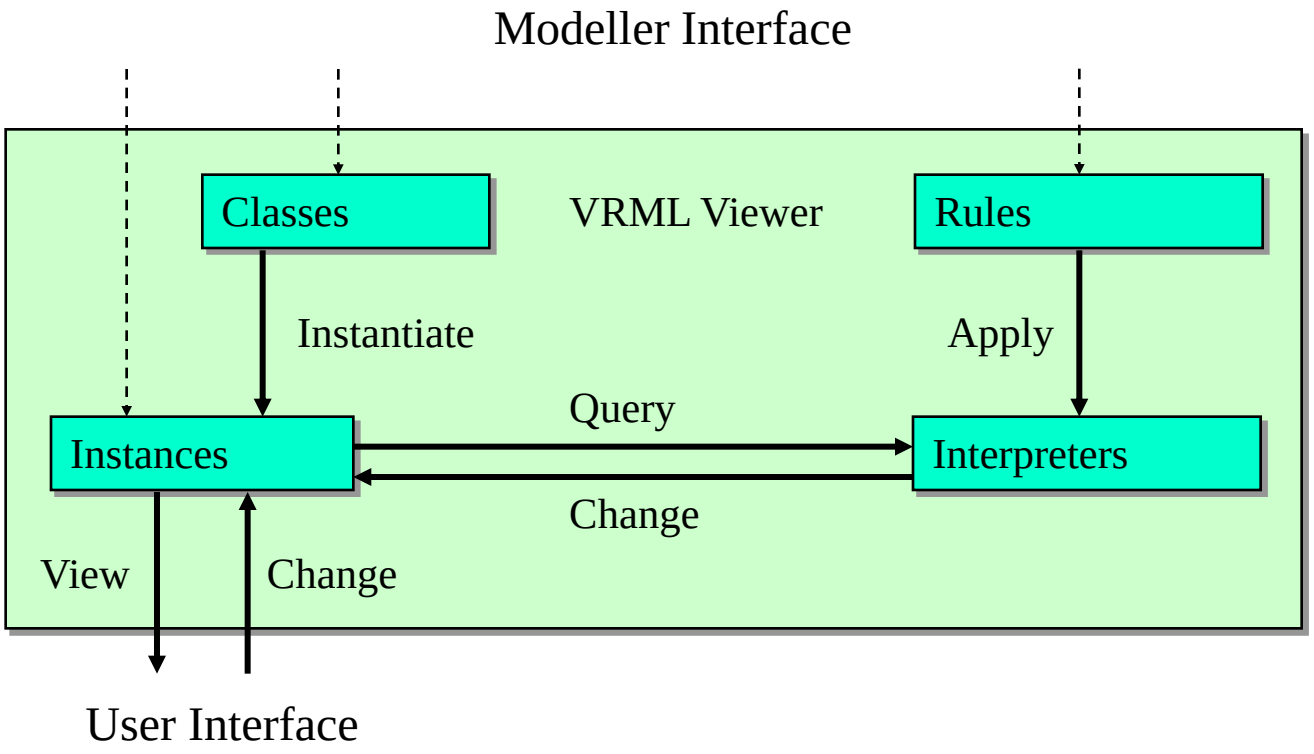
- Object classes as predefined frames
- Object interfaces with parameters
- Object instances of object classes
- Rules as logic dependencies between objects
- Rule interpreter as activator of the rules

Annotation: Implementation on the basis of existing VRML components
no predefined VRML components

Simple technology to specify

- generic static object features
- generic dynamic object behaviours
- interfaces to external systems

Object/Rule-Based Execution Model



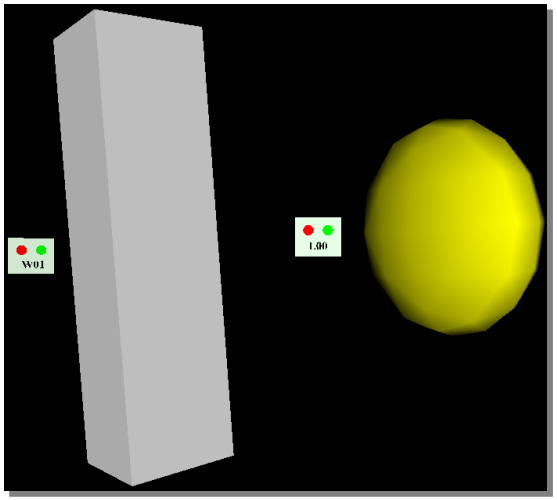
Object Classes

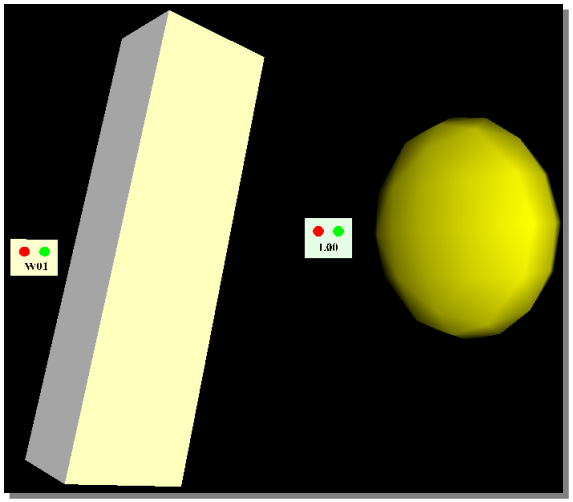
Object classes: Use of prototypes to define generic objects

Components:
- Class name
- Interface with parameters and types
- Default values for parameters
- Body with nodes and routes

```
PROTO ClassName [Interface] {Body}
```

Annotations:
- Body contains representations for different activation states of the object
- Body contains switchboard for manual changing the activation state of the object





Example:

```
PROTO WINDOW[
        exposedField SFColor material 1 1 1
        exposedField SFVec3f translate 0 0 0
        exposedField SFRotation rotate 0 1 0 0
        exposedField SFVec3f scale 1 1 1
        exposedField SFVec3f bbox 1 1 1
        field MFString name "W00"
        field MFString reference "W00.wrl"
        exposedField SFInt32 on 0     ]
{Transform {
  translation IS translate rotation IS rotate children [
   Transform {children[
    DEF S1 Switch {choice [
     Transform {scale IS scale children [
      Shape {appearance Appearance {
               material Material {diffuseColor IS material}}
               geometry Box {size 1 2 1}}]}
     Transform {scale IS scale rotation 0 0 1 0.3 children [
       Shape {appearance Appearance {
               material Material {diffuseColor IS material}}
               geometry Box {size 1 2 1}}]} ]
       whichChoice IS on} ]}
  Transform {translation 1 0 0 scale 0.5 0.5 0.5 children [
    DEF S Switchboard {Text IS name} ]}]}
ROUTE S.Select TO S1.whichChoice }
```

Object Interfaces

Object interfaces: Use of external prototypes to specify parameters of object classes

Components:
- Class name
- Interface with parameters and types
- URL with reference to prototype definition

```
EXTERNPROTO ClassName [Interface] [URL]
```

Annotations:
- Typical parameters are *material*, *translate*, *rotate*, *scale*, *bbox*, *name*, *reference*
- Interface contains one parameter *on* representing the activation state of the object
- Any number of other parameters is possible

Example:

```
EXTERNPROTO WINDOW [
        exposedField SFColor material
        exposedField SFVec3f translate
        exposedField SFRotation rotate
        exposedField SFVec3f scale
        exposedField SFVec3f bbox
        field MFString name
        field MFString reference
        exposedField SFInt32 on ]
["Prototypes.wrl#WINDOW"]
```

Object Instances

Object instances: Call of prototypes with concrete parameter values

Components: - Object name
- Class name
- Parameters and respective values according to the object interface

```
DEF ObjectName ClassName {ParameterValueList}
```

Annotations:
- Keyword parameters and values
- Parameter omission means use of default values
- Use in any appropriate VRML context

Example:

```
DEF Objects Transform {children [
 DEF W00 WINDOW {
  material 1 0 0 translate 0 -2 0 rotate 1 1 1 1.0 scale 1 1 1
  bbox 1 1 1 name "W00" reference "W00.wrl" on 0 }
 DEF L00 LIGHT {
  material 1 1 0 translate 4 0 0 rotate 0 1 0 0 scale 0.5 1 1
  bbox 0.5 1 1 name "L00" reference "L00.wrl" on FALSE }
 ] }
```

Rules

Rules: Definition of production rules for dependencies between object parameters

Components: - Condition
- Action

```
if (Condition) {Action}
```

Annotations:
- Use of JavaScript conditional clauses
- Condition: expressions with &&, ||, !, ==, <, >, !=, …
- Action: statements with =, …
- Arguments: object parameters, constants, variables, functions, …

Example:

```
if (W00.on == 1) {L00.on = TRUE; W00.on = 0;}
if (L00.on == TRUE) {V00.on = TRUE;}
if (D02.on == 1) {D02.on = 0;}
if (D00.on == 1 && D01.on == 1) {L01.on = TRUE;}
```

Rule Interpreters

Rule interpreters: Evaluation framework for rules in the context of objects

Components: - Script node with interface and control
- Rules

```
DEF Rules Script {ScriptInterface url "… Rules …"}
```

Annotations:
- Interface with declaration of all objects to be involved
- Control cycle for rule activation

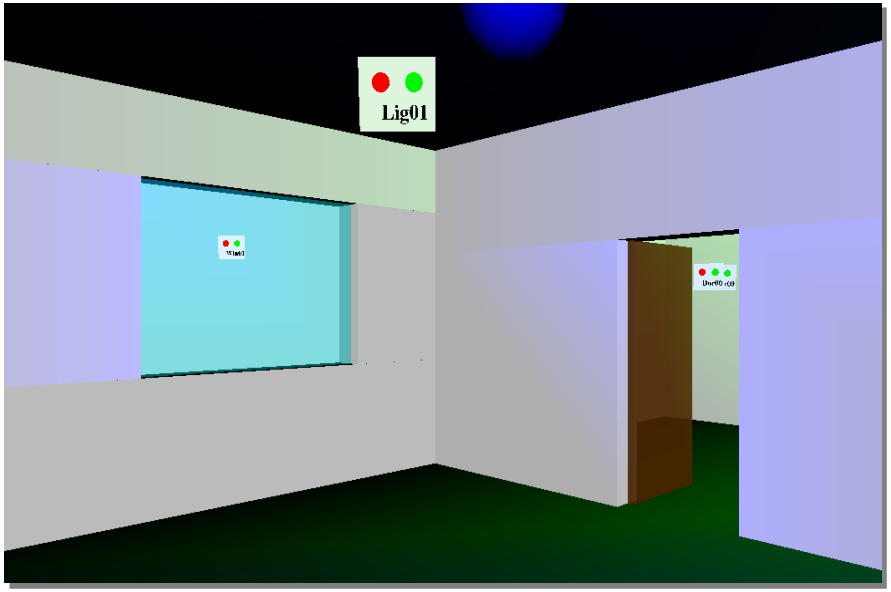
Example:

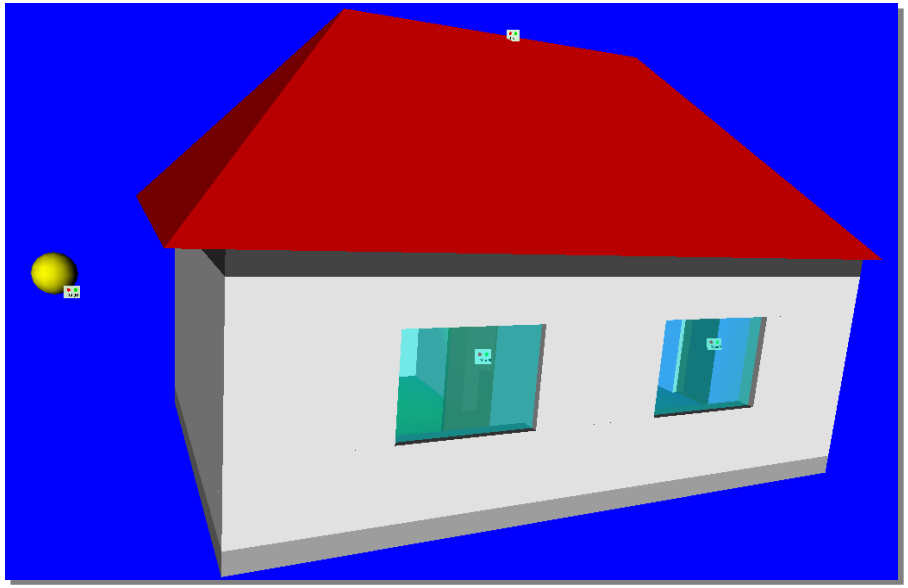
```
DEF Rules Script {
  eventIn SFTime Operation
  field SFNode W00 USE W00
  field SFNode L00 USE L00
  directOutput TRUE
  url "javascript:
    function Operation (wert,zeitmarke) {
        for (i=0;i<10;i++) { … } } " }
DEF Chronos0 TimeSensor {
 cycleInterval 4 loop TRUE startTime 1 stopTime 0}
ROUTE Chronos0.cycleTime TO Rules.Operation
```

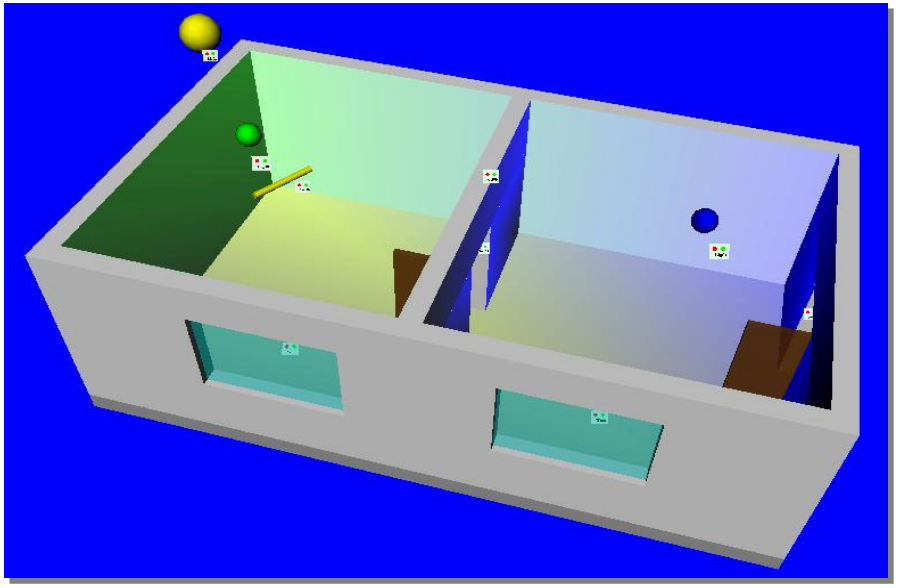
Examples

House as virtual intelligent environment with:

- Exterior structure
- Interior structure
- Function facilities
- Interaction facilities
- Navigation facilities







10. Applications of Virtual Intelligent Environments

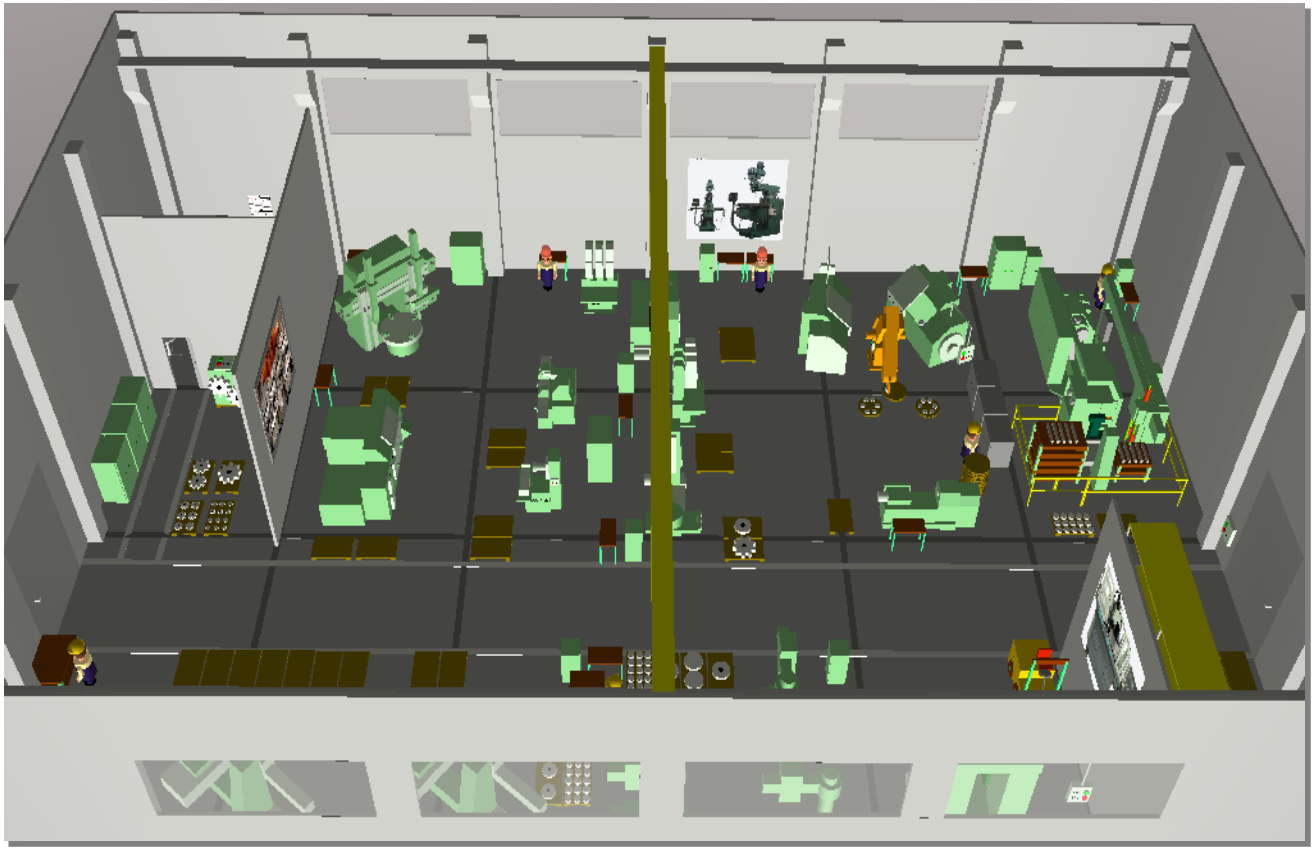
Prof. Dr.-Ing. habil. Wolfgang Oertel

Contents:

VIE for Campus Information

Oertel, W.:
Multilayer Spatiotemporal User Interface of a Campus Model.
In: Institut für Multimediatechnik (Hrsg.): 2. Kongress Multimediatechnik Wismar 2007.
Verlag Werner Hülsbusch, Boizenburg, 2007, S. 176-191

VIE for Manufacturing Planning



Oertel, W.; Görner, M.:
3D Modelling of Manufacturing Workshops Using VRML.
In: Gesellschaft zur Förderung angewandter Informatik (Hrsg.): 10. Anwendungsbezogener Workshop zur Erfassung, Modellierung, Verarbeitung und Auswertung von 3D-Daten (3D-NordOst 2007).
GfaI, Berlin, 2007, S. 147-154

VIE for Traffic Monitoring

Oertel, W.; Dimter, T.; Szoska, D.:
A Video-Based Approach for Stationary Platform Supervision.
In: The IEEE 5th International Conference on Intelligent Transportation Systems.
IEEE, ITSC, Singapore, 2002, S. 892-897

The End