1. VRML information

- The VRML repository:
  http://www.web3d.org/vrml/vrml.htm
- Browsers and plug-ins: z.B. cosmo player
  http://www.cai.com/cosmo/home.htm
- Books (example):

---

VRML-models in the www

- http://www.web3d.org/vrml/oblib.htm
- http://home.t-online.de/home/kiwano6/models.htm
VRML Files

ASCII Files: `<my_world>.wrl`

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

**Routes**

 utf8 is an international character set standard. It stands for: UCS (Universal Character Set) Transformation Format, 8-bit. Encodes 24,000+ characters for many languages, ASCII is a subset

Object description in VRML

**Nodes** (in an acyclic graph) with

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

**Examples:**

- **Shape nodes**
  
  Shape {appearance SFNode, geometry SFNode}

- **Geometry nodes**
  
  Cylinder {height SFFloat, radius SFFloat}

- **Appearance nodes**
  
  Appearance {material SSFNode, texture ...}
Minimal VRML-worlds

```vrml
#VRML V2.0 utf8
# generic cylinder
Shape {
  geometry Cylinder {
  } # Cylinder
} # Shape

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

# colored cylinder of variable size
# with missing pieces
Shape {
  appearance Appearance {
    material Material {
      diffuseColor 0.5 0.5 0.0
    } # material Material
  } # appearance Appearance
```
More geometry nodes

- Box \{\text{size \ldots}\}
- Cone \{\text{bottomRadius \ldots, height \ldots, \ldots}\}
- Sphere \{\text{radius \ldots}\}
- Text \{\text{string ["\ldots", "\ldots", \ldots], fontStyle \ldots, \ldots}\}

3. combination of objects

- Inlining of files
  Inline \{\text{url [\ldots, \ldots], \ldots}\}
- geometric Transformationen
  Transform \{
    \text{children [\ldots]}
    \text{translation \ldots \ldots \ldots}
    \text{Rotation \ldots \ldots \ldots}
    \text{Scale \ldots \ldots \ldots}
  \}
Example: sphere and cylinder

```
#VRML V2.0 utf8

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

# 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: wooden parts (Baufix)

```
#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
...
4. Reuse of models

- Inlining
- DEF und USE

**Multiple use of objects**

DEF Green-Box3

Inline {url 
"../blocks-
world/green-
box3.wrl"
} # Inline cylinder

USE Green-Box3

**Multiple use of object groups**

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

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
  }

Examples
CS in virtual worlds

- world CS
- object(s)
- camera(s)

5. Viewpoints

- Position of virtual camera with respect to the embedding CS

Viewpoint {
  description "initial camera position"
  position  0.0  0.0  25.0
  orientation  0.0  0.0  1.0  0.0
  fieldOfView  0.785 # 45 degrees
}
avatar navigation model

- movement types
  - Walk
  - Fly
  - Examine
  - NONE

- Speed
- Size
- Headlight
- Frustum

```
NavigationInfo {
  type "EXAMINE"
  headlight TRUE
  visibilityLimit 0.0
  avatarSize [0.25 1.6 0.75]
}
```

6. Animation in dynamic worlds

- Sensors: “Sources” of events
  - TimeSensor
  - Interactive sensors
  - Collision detection

```
TimeSensor {
  enabled TRUE
  startTime 0.0
  stopTime 0.0
  cycleInterval 1.0
  loop TRUE
  fraction_changed time
}
```
Movement Interpolationen (keyframe animation)

- VRML: linear interpolation between n key frames

```
PositionInterpolator {
  key [0.0 0.3 0.55]
  keyValue [
    0.0 0.0 0.0
    1.0 0.0 0.0
    3.5 3.5 0.0
  ]
  set_fraction
  value_changed
}
```

```
OrientationInterpolator {
  key [0.55 0.6 1.0]
  keyValue [
    0.0 0.0 1.0 0.0
    0.0 0.0 1.0 1.571
    0.0 0.0 1.0 6.284
  ]
  set_fraction
  value_changed
}
```

7. Routing between fields

- Route: connection between object node fields to propagate events

```
DEF T TimeSensor {
  enabled
  fraction_changed
}
```

```
DEF P PositionInterpolator {
  key [...]
  keyValue [...]
  set_fraction
  value_changed
}
```

```
DEF O Transform {
  translation
  rotation ...
  children [
    shape {
      geometry ...
    } ]
}
```

```
ROUTE T.fraction_changed TO P.set_fraction
ROUTE P.value_changed    TO O.set_translation
```

8. Anchor points

- Connection between two virtual worlds are established by anchor points
- (e.g., dungeon worlds, game levels)

```javascript
Anchor {
  url ["...", ...]
  children
  [...Box(...)]
  description "...
  ...
}
```

Anchors in AR-applications

- “hot-spots”
  - in 3D Menus (e.g., beside the object “at the wall”) for a dynamic, time-dependent help
  - in virtual models to augment specific areas with information
  - in virtual models to switch between alternative object views
  - in reality to visit alternative camera positions
9. Sensors

- **Sensors for different interaction types:**
  - **Pointer position (TOUCH):** isOver, isActive
    
    ```
    DEF Touch-Green TouchSensor{}
    ROUTE Touch-Green.touchTime TO Clock.set_startTime
    ```
  - **Pointer motion (MOTION):**
    - CylinderSensor
    - PlaneSensor
    - SphereSensor
    
    ```
    DEF Spin-Cylinder SphereSensor{}
    ROUTE Spin-Cylinder.rotation_changed TO Red-Cylinder2-b.set_rotation
    ```
  - If sensors are embedded, the most inner wins
  - Different incrementation modes (offsets)

10. Scripts

- **Connection VRML -> Java to enable more complex simulations**
- **Movement paths of objects (e.g., to realize physical object behaviors):**
  - Intelligent object responses
  - Communication in distributed systems (e.g., games with multiple players)
Scripts

- Example: Movement 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
}
```

VRML

Interactive demonstration

Examples