Virtual Reality Modeling Language

Thomas Jung

t.jung@fhtw-berlin.de

Motivation

- VRML ist unabhängiger Standard für 3D-Grafik im Internet
- VRML97 ist einziges offenes Dateiformat, das Animation und Interaktion beschreiben kann
- Modernisierung im Rahmen von X3D

Heute

- VRML – Entstehungsgeschichte
- Funktionalität von VRML 97
  - Geometriebeschreibung
  - Attribute
  - Animation
  - Interaktion
- Zukunft von VRML: X3D

Entstehungsgeschichte: VRML1.0

- 1. WWW-Konferenz, Genf, Frühjahr 1994
  - Gemeinsames Dateiformat für 3DGrafik im Internet
  - Name: VRML, Mailgroup für Anforderungen
- Herbst 1994: 1. Version eines Standards
  - Adaption eines bereits bestehenden Standards
  - OpenInventor-Format von Silicon Graphics
  - Autoren: Marc Pesce, Toni Parisi (Intervista) und Gavin Bell (SGI)

VRML1.0: Funktionalität

- Header: **#VRML V1.0 ascii**
- Funktionalität: Objekte, Lichtquellen, Materialien, Texturen, Kameras, Hierarchien, Transformationen
- Hyperlinks: WWWAnchor (clickable), WWWInline (automatisch)
- Besonderheiten: Instanziierung, Switch, LOD
- Erweiterbar

VRML1.0: Beleuchtung

- OpenGL-Beleuchtungsmodell
  - kleine Unterschiede
- Lowend-Renderingplattformen dürfen vereinfachtes Modell verwenden
  - Transparenzwerte nur 0 oder 1
  - Ambiente, spiegelnde und emissive Reflexion müssen nicht unterstützt werden
- Konsequenz: Welten sahen auf unterschiedlichen Plattformen unterschiedlich aus

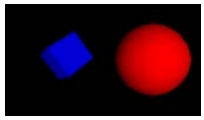
VRML 2.0 / VRML97

- Leichte Veränderung der Syntax
  - Tool: vrml1tovrml2
- Zusätzlich
  - Beschreibung von Animation (Bewegungsabläufe)
  - Beschreibung von Verhalten (Interaktion)

VRML - Authoring

- Standard beschreibt Dateiformat
- Darstellung in der Regel über Web-Browser-Plugin
  - Cosmo-, Cortona-, Blaxxunplayer
  - Auch Standalone-System möglich
  - Beides nennt man VRML-Browser
- VRML-Authoring mit 3D-Werkzeugen
  - 3DSMax, Maya, etc.
  - Früher VRML-Spezialisten wie CosmoWorlds

VRML - Beispiel



```
#VRML V2.0 utf8
Transform {
  children [
    NavigationInfo { headlight FALSE } # We'll add our own light

    DirectionalLight {      # First child
      direction 0 0 -1      # Light illuminating the scene
    }

    Transform {     # Second child - a red sphere
      translation 3 0 1
      children [
        Shape {
          geometry Sphere { radius 2.3 }
          appearance Appearance {
            material Material { diffuseColor 1 0 0 }
                                        # Red
          }
        }
      ]
    }
    ...
```

```
    ...
    Transform {         # Third child - a blue box
      translation -2.4 .2 1
      rotation      0 1 1  .9
      children [
        Shape {
          geometry Box {}
          appearance Appearance {
            material Material { diffuseColor 0 0 1 }
                                                #Blue
          }
        }
      ]
    }

  ] # end of children for world
}
```


VRML97-Spezifikation

- *http://www.vrml.org/technicalinfo/specifications/vrml97*
- Nach dem Header kommen:
  - Knoten
    - Transform { ... }
    - **DEF** *Bla* Transform { ... } Benennung von Knoten!
    - **USE** *Bla* Referenzierung von Knoten
  - Route-Befehle
    - **ROUTE** *BlaNode.blaEventOut* **TO** *Bla2Node.blaEventIn* Animation / Interaktion
  - Prototypen
    - **PROTO** *BlaTyp* [...] { ... } „Klassendefinitionen"
    - **EXTERNPROTO** *BlaTyp* [...] [„url1" | „url2"] ... in Dateien

VRML - Knoten

- Spezifikation:

  Nicht bei eventIn/Out

```
KnotenTypName {
        Feldtyp Datentyp FeldName DefaultWert Wertebereich
}
```

- Bei Benutzung nur Feldnamen und -werte

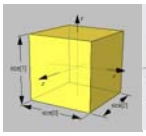
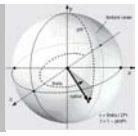
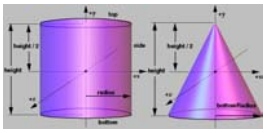
```
Cone {
    bottomRadius 1
    height 2
    side TRUE
    bottom TRUE
}
```


VRML - Datentypen

SFBool	(kein MFBool!!!)	TRUE, FALSE
SFFloat		Gleitkommazahl
SFTime		**SFFloat**
SFVec2f		**SFFloat SFFloat**
SFVec3f		**SFFloat SFFloat SFFloat**
SFColor		**SFFloat SFFloat SFFloat**
SFRotation		**SFFloat SFFloat SFFloat SFFloat**
SFInt32		Ganze Zahl
SFImage	(kein MFImage!!!)	**SFInt32 SFInt32** ...
SFNode		Knoten, NULL
SFString		„..."
MFxxx		[**SFxxx SFxxx ...**], [ ], **SFxxx**

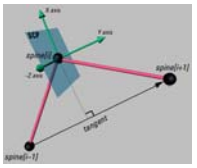
VRML - Primitivobjekte

```
Box {
  field  SFVec3f size  2 2 2      # (0, unendl.)
}

Sphere {
  field  SFFloat radius  1    # (0,unendl.)
}

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

Cylinder {
  field   SFBool    bottom  TRUE
  field   SFFloat   height  2          # (0,unendl.)
  field   SFFloat   radius  1          # (0,unendl.)
  field   SFBool    side    TRUE
  field   SFBool    top     TRUE
}
```

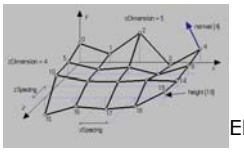




Komplexere Objekte

- Listenobjekte
  - IndexedFaceSet { ... }
  - IndexedLineSet { ... }
  - PointSet { ... }



Extrusion { ... }



ElevationGrid { ... }

Beispiel Extrusion

```
Shape {
    appearance Appearance {
        material Material { diffuseColor 1 0 0 }
    }
    geometry Extrusion {
        crossSection [1 1, 1 -1, -1 -1, -1 1, 1 1]
        spine [0 0 0, 0 2 0, 1 3 0, 2 3 0]
        scale [1 1, 1 0.5, 0.5 1, 0.5 0.5]
    }
}
```

Hinweise zur Form von Objekten

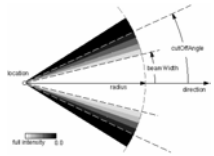
- Bei ElevationGrid, IndexedFaceSet, Extrusion, ...
- creaseAngle
  - Winkel zw. Normalen > creaseAngle ⇒ Eckpunktnormalen sind Flächennormalen
- solid
  - TRUE, dann kein Backfacing
- ccw
  - TRUE, dann Normale gemäß Rechter-Hand-Regel
  - Wenn Normalen explizit angegeben werden, sollten sie zum Wert passen
- convex
  - TRUE, dann Polygonecken in einer Ebene und Winkel kleiner als 180 Grad,
  - FALSE ist problematisch !!!!

Materialien

```
Shape {
    SFNode appearance
    SFNode geometry
}

Appearance {
    SFNode material
    SFNode texture
    SFNode textureTransform
}

Material {
    SFFloat ambientIntensity
    SFColor diffuseColor
    SFColor emissiveColor
    SFFloat shininess
    SFColor specularColor
    SFFloat transparency
}
```

```
ImageTexture {
        MFString url
        SFBool   repeatS
        SFBool   repeatT
}
MovieTexture {
        SFBool   loop
        FFloat   speed
        SFTime   startTime
        SFTime   stopTime
        MFString url
        SFBool   repeatS
        SFBool   repeatT
        SFTime   duration_changed
        SFBool   isActive
}
PixelTexture {
        SFImage  image
        SFBool   repeatS
        SFBool   repeatT
}
```


Beleuchtung



```
DirectionalLight {
    SFFloat ambientIntensity
    SFColor color
    SFVec3f direction
    SFFloat intensity
    SFBool  on
}

PointLight {
...wie oben, jedoch ohne direction,
   dafür zusätzlich...
    SFVec3f location
    SFVec3f attenuation
    SFFloat radius
}
```

```
SpotLight {
... Wie PointLight, zusätzlich ...
    SFFloat beamWidth
    SFFloat cutOffAngle
    SFVec3f direction
}

NavigationInfo {
    SFBool headlight
}
```

Animation

```
TimeSensor {
  exposedField SFTime   cycleInterval
  exposedField SFBool   enabled
  exposedField SFBool   loop
  exposedField SFTime   startTime
  exposedField SFTime   stopTime
  eventOut     SFTime   cycleTime
  eventOut     SFFloat  fraction_changed
  eventOut     SFBool   isActive
  eventOut     SFTime   time
}
```

```
ColorInterpolator {
  eventIn       SFFloat set_fraction
  exposedField MFFloat key
  exposedField MFColor keyValue
  eventOut      SFColor value_changed
}
```

- CoordinateInterpolator
- NormalInterpolator
- OrientationInterpolator
- PositionInterpolator
- ScalarInterpolator

- AudioClip und MovieTexture benötigen keine ROUTE

```
ROUTE <name>.<field/eventName> TO <name>.<field/eventName>
```


VRML – Feldtypen

(Namen geändert in X3D!)

- Wichtig für Routing in VRML !!!

- **field** Bla: **nicht verknüpfbar**
- **eventIn** Bla: set_Bla = xxx
- **eventOut** Bla: xxx = Bla_changed
  (für Boolsche Felder) isBla
- **exposedField** ist eventIn plus eventOut !

- (Außerdem: addChildren, removeChildren)

Beispiel: Animation

```
#VRML V2.0 utf8
Inline {  url ["axis.wrl"]}
DEF  BALL Transform{  children [  Shape {  appearance Appearance {
          material Material {  diffuseColor 1 0 0 } }
          geometry Sphere { radius .25 } } ] }
DEF  PATH PositionInterpolator {  key [  0   .25   .5   .75   1  ]
          key  keyValue [  -3  0  0,   0  3  0,   3  0  0,   0 -3  0,  -3  0  0, ] }
DEF  TIMER TimeSensor {  loop TRUE  cycleInterval 5}
ROUTE TIMER.fraction_changed TO PATH.set_fraction
ROUTE PATH.value_changed TO BALL.translation
```

Verhalten/Interaktion

- Umgebungssensoren, abhängig vom Standpunkt
  - Collision
  - ProximitySensor
  - TimeSensor
  - VisibilitySensor

- Sensoren, die durch die Mouse aktiviert werden
  - Anchor
  - CylinderSensor
  - PlaneSensor
  - SphereSensor
  - TouchSensor

- **Scripting** (JAVA, JAVAScript oder VRMLscript)

TouchSensor

PlaneSensor

CylinderSensor

SphereSensor

Weitere Knoten

- Spezielle Objekte: Background, **Billboard,** Text, Sound, Fog
- WorldInfo, NavigationInfo
- Gruppierung von Knoten: Anchor, Billboard, Collision, Group, Inline, LOD, Switch, Transform
- Wiederverwendung: DEF, USE, PROTO

Billboard

Transformationen

- P' = **T** × **C** × **R** × **SO** × **S** × -**SO** × -**C** × P

```
Transform {
    MFNode      addChildren
    MFNode      removeChildren
    MFNode      children
    SFVec3f     bboxCenter
    SFVec3f     bboxSize
    SFVec3f     translation
    SFVec3f     center
    SFRotation  rotation
    SFRotation  scaleOrientation
    SFVec3f     scale
}
```

```
Viewpoint {
    SFBool      set_bind
    SFFloat     fieldOfView
    SFBool      jump
    SFRotation  orientation
    SFVec3f     position
    SFString    description
    SFTime      bindTime
    SFBool      isBound
}
```

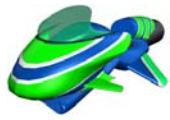
- Anspringen über Browser-Menu,
- Binden an Animation möglich

Level of Detail

- **LOD {**MFNode **level []**
  SFVec3f **center 0 0 0**
  MFFloat **range []**
  }

1	2.3	7.9	25	*(Entfernung)*
Genau	weniger grob	primitiv	unsichtbar	*(Knoten)*

- Auslagerung von Leveln durch Anchor-Knoten für Optimierung der Ladezeit
- visuelle Übergänge sollten nicht erkennbar sein !!



VRML - Prototypen

Definition

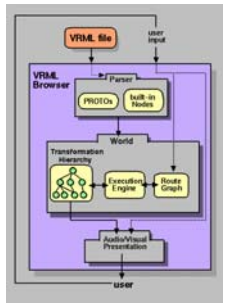
```
PROTO SphereCone [ field SFFloat radius 2.0
                   field SFFloat height 5.0
                   field SFNode sphereApp NULL
                   field SFNode coneApp NULL ]
{ Transform { children [ Shape { appearance IS sphereApp
                                 geometry  Sphere { radius IS radius } }
                         Shape { appearance IS coneApp
                                 geometry Cone { height IS height } } ] }}
```

```
Transform{ translation 15 0 0
           children SphereCone { radius 5.0       height 20.0
           sphereApp Appearance { material Material { ... } }
           coneApp Appearance { texture ImageTexture { ... } } }}
```

Benutzung

VRML - Browser

- VRML-Autorensystem
  - z. B. CosmoWorlds
  - liefert VRML-Szene
- VRML-Browser
  - z. B. CosmoPlayer
  - interpretiert VRML-Szene



External Authoring Interface

- Zur Kommunikation zwischen VRML-Browser und Applikation
- Erster Vorschlag 1996/97
  - JAVA-basierte Implementierung von SGI
- Dezember 1997 standardisierte sprachunabhängige Version
  - nicht Bestandteil von VRML 2.0
- JAVA-Sicht auf VRML-Welt im Browser (Applet)

```
... getBrowser() ...
BrowserNode mover = browser.getNode("Mover");
EventInSFVec3f translation =
  (EventInSFVec3f) mover.getEventIn("set_translation");
float value[3] = new float[3];
value[0] = 5; value[1] = 0; value[2] = 1;
translation.setValue(value);
```


VRML - Konkurrenten

- JAVA3D
  - Programmierschnittstelle, kein Szenenformat
- XML
  - Nicht direkt für 3D-Szenen, -> X3D
- **Director-3D**
- Produkte/Demos
  - Active Worlds, Viscape, Meme, Chrome, ...
- DirectX-Dateiformat

VRML - Wie geht´s weiter

- Web3D
  - X3D - Encoding basierend auf XML
- MPEG4
  - 3D - Anteile in X3d-Syntax
- JAVA3D
  - Szenenbeschreibung in VRML/X3D möglich
  - Loader für VRML97/X3d

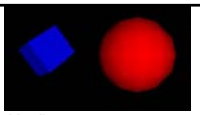
X3D

- XML-basiertes „VRML 3.0“
- Spezifikation wird zur Zeit verabschiedet
- Exporter für Maya, 3dsmax, etc. schon verfügbar
- Kern ist kompakter als bei VRML97
- VRML97-Dateien können meist weiterverwendet werden
  - nur geringfügige Änderungen bei VRML-Encoding

Vorteile von X3D

- Erweiterter Szenengraph
  - Mehr Knoten wie z.B. NURBS und Multitexturing
- Besseres Programmiermodell
  - statt EAI und Scriptknoten einheitliches Modell
- Mehrere File Encodings
  - Einerseits XML andererseits klassisches VRML
- Modulare Architektur
  - Kleinere Kernfunktionalität plus Anwendungsspezifische Erweiterungen
  - Saubere Erweiterungsmechanismen

XML-Encoding



```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE X3D PUBLIC "ISO//Web3D//DTD X3D 3.0//EN"
"http://www.web3d.org/specifications/x3d-3.0.dtd">
<X3D version='3.0' profile='Interchange'>
 <head>
   <meta name='filename' content='RedSphereBlueBox.x3d'/>
 </head>
 <Scene>
  <Transform>
    <NavigationInfo headlight='false,
         avatarSize='0.25 1.6 0.75'
         type='"EXAMINE"'/>
    <DirectionalLight/>
    <Transform translation='3.0 0.0 1.0'>
     <Shape>
       <Sphere radius='2.3'/>
       <Appearance>
         <Material diffuseColor='1.0 0.0 0.0'/>
       </Appearance>
     </Shape>
    </Transform>
        ...
                                    ...
                <Transform translation='-2.4 0.2 1.0'
                     rotation='0.0 0.707 0.707 0.9'>
                  <Shape>
                  <Box/>
                  <Appearance>
                    <Material diffuseColor
                        ='0.0 0.0 1.0'/>
                  </Appearance>
                 </Shape>
                </Transform>
              </Transform>
            </Scene>
           </X3D>
```


Zusammenfassung

- VRML ist offener Standard
- Primitivobjekte, Extrusionen, ElevationGrids und Listen
- Beleuchtung und Texturierung angelehnt an OpenGL
- Animation und Interaktion
- VRML-Konkurrenz
- X3D