

Knoten zur Beschreibung des graph. Szenengraphen

- Alle Geometrie-Knoten müssen Kind eines **Shape**-Knotens sein
- Definition:


```
Shape {
    SFNode geometry NULL
    SFNode appearance NULL
}
```
- Achtung:
 - Das Feld **geometry** darf nur Geometrie-Knoten enthalten (es gibt etliche Klassen von Geometrie-Knoten)
 - Das Feld **appearance** darf nur einen Appearance-Knoten enthalten (es gibt nur eine Klasse von Appearance-Knoten)
- **Shape**-Knoten dienen dazu, Geometrie mit einer **Appearance** zu verknüpfen

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 X3D / VRML97 20

- Der Appearance-Knoten dient zur Spezifikation des Aussehens einer Geometrie
- Definition:


```
Appearance {
    SFNode material NULL
    SFNode texture NULL
    SFNode fillProperties NULL
    ...
}
```
- Auch hier gilt wieder: die Werte eines Feldes (hier: Instanzen einer Knotenklasse) müssen vom "richtigen" Typ (d.h., der richtigen Klasse) sein

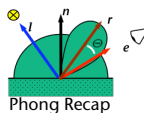
G. Zachmann Virtuelle Realität und Simulation - WS 09/10 X3D / VRML97 21

■ Der Material-Knoten:

```

Material {
  SFFloat ambientIntensity 0.2
  SFCOLOR diffuseColor      0.8 0.8 0.8
  SFCOLOR emissiveColor     0 0 0
  SFCOLOR specularColor     0 0 0
  SFFloat shininess         0.2
  SFFloat transparency      0
}

```



Phong Recap

■ Der Textur-Knoten:

```

ImageTexture {
  MFString url      [ ]
  SFBBool repeats  TRUE
  SFBBool repeatT  TRUE
}

```

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 X3D / VRML97 22

■ Auf der Homepage der Vorlesung finden Sie unter "*Online Literatur und Resources im Internet*" ein großes Archiv mit Materialien

■ Beispiele:




■ Weitere Resource: ein Material-Editor (in Java)
<http://tog.acm.org/resources/applets/vrml/pellucid.html>

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 X3D / VRML97 23

Ein erstes Beispiel

```
#X3D V3.1 utf8
Shape {
  geometry Cone {
    bottomRadius 1
    height 2
  }
  appearance Appearance {
    material Material {
      ambientIntensity 0.256
      diffuseColor 0.029 0.026 0.027
      shininess 0.061
      specularColor 0.964 0.642 0.980
    }
  }
}
```

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 X3D / VRML97 24

Geometrie-Knoten für Terrain

- Allgemein für (diskrete) Flächen, die sich als Funktion über einer Ebene beschreiben lassen
- Definition:

```
ElevationGrid {
  SFFloat normalPerVertex TRUE
  SFFloat creaseAngle 0.0
  MFFloat height []
  SFInt32 xDimension 0
  SFFloat xSpacing 1.0
  SFInt32 zDimension 0
  SFFloat zSpacing 1.0
  SFFloat solid TRUE
}
```

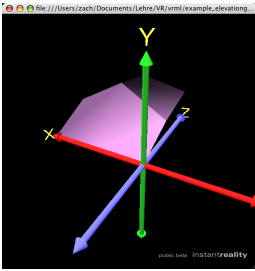
G. Zachmann Virtuelle Realität und Simulation - WS 09/10 X3D / VRML97 25

- Bedeutung der Felder:
 - **normalPerVertex** schaltet Beleuchtung pro Vertex mit Gouraud-Shading ein (die Normalen werden i.A. vom Browser berechnet)
 - **solid = TRUE** schaltet Backface-Culling ein
 - Tip: bei Terrain ausschalten
 - Alle Winkel zwischen 2 Polygonen über eine Kante hinweg (*dihedral angle*), die größer als **creaseAngle** sind, werden erhalten, d.h., die beteiligten Vertices werden für die beiden Polygone mit jew. einer eigenen Normale gerendert
- Anmerkungen:
 - Aus Matlab kann man Plots als ein solches VRML-ElevationGrid exportieren
 - Achtung: die Vierecke sind i.A. nicht planar → Flackern und andere Artefakte!

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 X3D / VRML97 26

- Beispiel:

```
Shape {
  appearance Appearance { ... }
  geometry ElevationGrid {
    height [
      0.0 0.0 0.0
      0.2 0.5 0.2
      0.3 0.4 0.1 ]
    xDimension 3
    zDimension 3
    xSpacing 0.5
    zSpacing 0.5
    solid false
    #creaseAngle 1.5
  }
}
```



[example_elevationgrid.wrl](#)

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 X3D / VRML97 27

Dreiecke

- Die allgemeinste Geometrie
- Für Dreiecke (und Vierecke) gibt es viele Varianten; hier nur 2
- Die einfachste Variante: **TriangleSet**
- Definition:

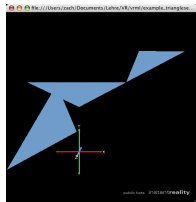

```
TriangleSet {
  SFNode coord      NULL
  SFFloat ccw       TRUE
  SFFloat normalPerVertex TRUE
  SFFloat solid     TRUE
  SFFloat creaseAngle 0.0
}
```

```
Coordinate {
  MFVec3f point []
}
```
- coord** → **point** ist eine Liste von Koordinaten; je 3 aufeinanderfolgende ergeben einen Vertex; davon je 3 aufeinanderfolgende ergeben ein Dreieck
- ccw** (*counter-clockwise*) gibt an, ob die Vertices im Uhrzeigersinn vorliegen oder nicht

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 X3D / VRML97 28

- Beispiel:


```
Shape {
  appearance Appearance { ... }
  geometry TriangleSet {
    coord Coordinate {
      point [ -2 0 3, -0 1 1, -1 3 0,
              0 2 0, 2 3 1, -2 3 1,
              3 5 -2, 2 3 1, 4 4 2 ]
    }
    solid FALSE
    ccw TRUE
  }
}
```



[example_triangleset.wrl](#)
- Bemerkung:
 - das Komma ist in X3D/VRML ein *Whitespace*
 - könnte man also weglassen; sollte man bei hand-geschriebenen Szenen aber nicht

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 X3D / VRML97 29

Ein häufig vorkommender Knoten ist **IndexedTriangleSet**:

```
IndexedTriangleSet {
  SFNode coord NULL
  MFInt32 index []
  SFBool ccw TRUE
  SFBool normalPerVertex TRUE
  SFBool solid TRUE
  SFFloat creaseAngle 0.0
}
```

```
Coordinate {
  MFVec3f point []
}
```

Großer Vorteil: Speicher-Einsparung
 Denn: bei "normalen" Dreiecks-Meshes wird jeder Vertex im Schnitt von 6 Dreiecken "benutzt"

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 X3D / VRML97 30

Dasselbe Beispiel nochmal, diesmal mit **IndexedTriangleSet**:

```
Shape {
  appearance Appearance { ... }
  geometry IndexedTriangleSet {
    index [ 0 1 2, 3 4 5, 6 4 7 ]
    coord Coordinate {
      point [ -2 0 3, -0 1 1, -1 3 0,
              0 2 0, 2 3 1, -2 3 1,
              3 5 -2, 4 4 2 ]
    }
    solid FALSE
    ccw TRUE
  }
}
```

[example_indexedtriangleaset.wrl](#)

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 X3D / VRML97 31

- Der häufigste Knoten-Typ ist (unnötigerweise) das **IndexedFaceSet**:

```

IndexedFaceSet {
  SFNode coord NULL
  MFInt32 coordIndex []
  SFBool ccw TRUE
  SFBool normalPerVertex TRUE
  SFBool solid TRUE
  SFFloat creaseAngle 0.0
}

Coordinate {
  MFVec3f point []
}

```

- Unterschied zu **IndexedTriangleSet**: die -1 als "Sentinel"

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 X3D / VRML97 32

- Vorteil: beliebige Polygone
- Anmerkung: viele Exporter exportieren **IndexedFaceSet** obwohl alle Pgone Dreiecke sind → Speicherverschwendung & langsames Rendering!
- Das Beispiel von vorhin nochmal als **IndexedFaceSet**:

```

Shape {
  appearance Appearance { ... }
  geometry IndexedTriangleSet {
    coordIndex [ 0 1 2 -1 3 4 5 -1 6 4 7 -1 ]
    coord Coordinate {
      point [ -2 0 3, -0 1 1, -1 3 0,
              0 2 0, 2 3 1, -2 3 1,
              3 5 -2, 4 4 2 ]
    }
    solid FALSE
    ccw TRUE
  }
}

```

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 X3D / VRML97 33

Spezifikation weiterer Attribute pro Vertex

- In allen Geometrie-Knoten kann man weitere Vertex-Attribute spezifizieren, z.B., Normalen oder Texturkoord. pro Vertex
- Hier am Beispiel Texturkoord. im **IndexedFaceSet**:

```

IndexedFaceSet {
  SFNode coord
  MFInt32 coordIndex
  SFNode texCoord
  MFInt32 texCoordIndex
  SFBool ccw
  SFBool normalPerVertex
  SFBool solid
}

TextureCoordinate {
  MFVec2f point []
}

Coordinate {
  MFVec3f point []
}

```

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 X3D / VRML97 34

Weitere Geometrie-Knoten

- Es gibt noch viele weitere:
 - PointSet, LineSet, QuadSet, ...**
 - IndexedLineSet, IndexedQuadSet, ...**
 - TriangleStripSet, IndexedTriangleStripSet, ...**
 - Box, Sphere, Cylinder, ...**
 - Text, Extrusion, ...**
- Viele 2D-Knoten, z.B.: **Arc2D, Polyline2D, ...**
- CAD-Knoten: **CADAssembly, NurbsPatchSurface, ...**

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 X3D / VRML97 35

Knoten zur Hierarchie-Bildung

- Einfache Gruppen-Knoten:


```
Group {
  MFNode children []
}
```
- Die Knoten im Feld **children** dürfen wieder **Group**-Knoten sein oder **Shape**-Knoten
- Beispiel: ...

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 X3D / VRML97 37

- Transformationen:


```
Transform {
  MFNode children []
  SFVec3f center 0 0 0
  SFRotation scaleOrientation 0 0 1 0
  SFVec3f scale 1 1 1
  SFRotation rotation 0 0 1 0
  SFVec3f translation 0 0 0
}
```
- Alle Kinder unter einem **Transform**-Knoten werden transformiert
 - Oft hat ein **Transform**-Knoten nur 1 Kind
- Bedeutung:

$$p' = T \cdot C \cdot R_2 \cdot R_1 \cdot S \cdot R_1^{-1} \cdot C^{-1} \cdot p$$
 - scaleOrientation erlaubt also eine Skalierung entlang beliebiger (lokaler) Achsen, nicht nur entlang der lokalen Koord.achsen

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 X3D / VRML97 38

- Ein "include"-Mechanismus mittels des **InLine**-Knotens:


```

      InLine {
        SFFloat load TRUE
        MFString url []
      }
      
```
- Mit **load=FALSE** kann man das Laden der Teil-Szene aufschieben; bei **TRUE** wird die Teil-Szene beim Parsen der Parent-Szene geladen
- Die erste gefundene URL im Feld **url** wird genommen
- Beispiel:


```

      Transform {
        scale 0.5
        children [
          InLine {
            url [ "coordAxes.wrl"
                 "http://my.site.com/coordAxes.wrl" ]
          }
        ]
      }
      
```

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 X3D / VRML97 39

Ein einfacher Schalter

- Mit dem **Switch**-Knoten kann man eines aus mehreren Kindern einschalten
- Definition:


```

      Switch {
        SFInt32 whichChoice -1
        MFNode children []
      }
      
```
- **whichChoice=-1** schaltet alle Kinder ab, **whichChoice=0** schaltet das erste Kind an

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 X3D / VRML97 40

- Mit diesem Knoten kann man Hinweistafeln u.ä. erstellen:


```
Billboard {
  SFVec3f axisOfRotation 0 1 0
  MFNode  children      []
}
```
- Dieser Knoten erzeugt in jedem Frame eine Transformation, die dafür sorgt, daß die lokale z-Achse zum aktuellen Viewpoint zeigt
- **axisOfRotation** wird im lokalen Koordinatensystem spezifiziert
- Falls **axisOfRotation** = (0,0,0) ist, dann wird zusätzlich die lokale y-Achse parallel zur y-Achse des Viewers ausgerichtet

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 X3D / VRML97 41

Wiederverwendung von Szenengraphenteilen

- Beispiele, wo Wiederverwendung Sinn macht:
 - Ein Teil der Geometrie kommt mehrfach in der Szene vor (i.A. an verschiedenen Positionen)
 - Dieselbe Appearance (Material / Textur) soll auf verschiedene Geometrien angewendet werden
- Mechanismus in X3D/VRML:
 - Namen für einen Knoten definieren:

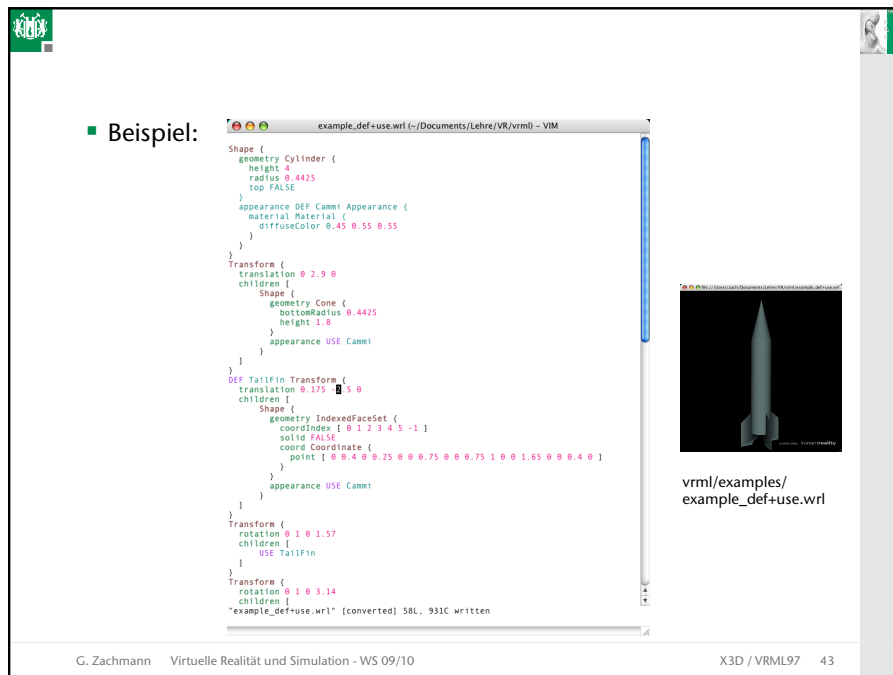

```
DEF NodeName NodeType {
  fields ...
}
```
 - An jeder Stelle, wo ein Knoten des entsprechenden Typs stehen kann, kann nun einfach


```
USE NodeName
```

 verwendet werden

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 X3D / VRML97 42

■ Beispiel:



```

Shape {
  geometry Cylinder {
    height 4
    radius 0.4425
    top FALSE
  }
  appearance DEF Cam1 Appearance {
    material Material {
      diffuseColor 0.45 0.55 0.55
    }
  }
}
Transform {
  translation 0 2.9 0
  children [
    Shape {
      geometry Cone {
        bottomRadius 0.4425
        height 1.8
      }
      appearance USE Cam1
    }
  ]
}
DEF TailFin Transform {
  translation 0.175 0 1.5 0
  children [
    Shape {
      geometry IndexedFaceSet {
        coordIndex [ 0 1 2 3 4 5 -1 ]
        solid FALSE
        coord Coordinate {
          point [ 0 0.4 0 0.25 0 0 0.75 0 0 0.75 1 0 0 1.65 0 0 0.4 0 ]
        }
      }
      appearance USE Cam1
    }
  ]
}
Transform {
  rotation 0 1 0 1.57
  children [
    USE TailFin
  ]
}
Transform {
  rotation 0 0 0 3.14
  children [
    "example_def+use.wrl" [converted] 58L, 931C written
  ]
}

```

vrml/examples/
example_def+use.wrl

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 X3D / VRML97 43

Bemerkungen

- Die Bezeichnung **DEF** ist sehr unglücklich
- Wahre Semantik / Eselsbrücke: **DEF** ≈ "Name", **USE** ≈ Pointer!
- Scope: reicht vom **DEF** bis zum Ende des Files — Klammern ({ } []) spielen keine Rolle!
- **DEF** muß im File **vor USE** kommen (logisch), aber nicht notwendigerweise auf demselben Level im Szenengraph
 - Dadurch könnte man sogar Zyklen im Graph erzeugen!
- Tip: sinnvolle Namen vergeben

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 X3D / VRML97 44

Der Verhaltensgraph

- "Animationen" (i.e., dynamische Szenengraphen) sind Veränderungen des Szenengraphen; z.B.:
 - Änderungen von Transformationen, z.B. die Position von Objekten oder die Bewegung eines Roboterarmes,
 - Änderungen des Materials, z.B. der Farbe oder der Texturkoord. eines Objektes,
 - Deformation eines Objektes, d.h., Änderungen der Vertex-Koord.,
- Alle diese Veränderungen sind äquivalent zur **Änderung eines Feldes eines Knotens zur Laufzeit**

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 X3D / VRML97 45

Events und Routes

- Der Mechanismus in X3D zur Veränderung des Szenengraphen:
 - Es gibt spezielle Knoten, deren Felder sich ändern
 - Eine Änderung eines Feldes erzeugt einen sog. **Event**
 - Felder können miteinander durch sog. **Routes** verbunden werden
 - Bei Auftreten eines Events wird der Inhalt des Feldes vom Route-Anfang zum Feld des Route-Endes kopiert ("der Event wird propagiert")

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 X3D / VRML97 46

- Felder haben nicht nur Typ und Wert, sondern auch einen sog. "access type":

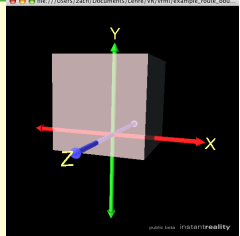
VRML97	X3D
eventIn	inputOnly
eventOut	outputOnly
field	initializeOnly
exposedField	inputOutput
- Felder mit einem Namen **zzz**, die den Access-Type exposedField haben, haben implizit den Namen **zzz_changed**, wenn sie als Ausgabe-Feld verwendet werden, und den Namen **set_zzz**, wenn sie als Eingabe-Feld verwendet werden
 - Viele der vordefinierten Felder in vordef. Knoten sind **exposedField**'s

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 X3D / VRML97 47

- Ein einfaches Beispiel:


```

DEF ts TimeSensor {
  loop TRUE
  cycleInterval 5
}
DEF pi PositionInterpolator {
  key [ 0 0.5 1 ]
  keyValue [ 0 -1 0, 0 1 0, 0 -1 0 ]
}
DEF tr Transform {
  translation 0 0 0
  children [
    Shape { geometry Box { } }
  ]
}
ROUTE ts.fraction_changed TO pi.set_fraction
ROUTE pi.value_changed TO tr.set_translation
      
```



[example_route_bounce.wrl](#)

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 X3D / VRML97 48

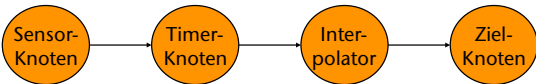
▪ Syntax der Routes:

```
ROUTE nodeName.outputFieldName TO nodeName.inputFieldName
```

- Die Knoten müssen früher im File mit **DEF** spezifiziert worden sein
- Routes dürfen nur von Ausgabe-Feldern (d.h., **eventOut** oder **exposedField**) zu Eingabe-Feldern (d.h., **eventIn** oder **exposedField**) gezogen werden
- Felder dürfen *fan-in* und *fan-out* haben (mehrere eingehende / ausgehende Routes)
 - Verhalten bei gleichzeitiger Ankunft mehrerer Events ist undefiniert!

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 X3D / VRML97 49

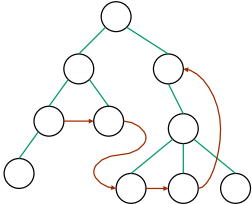
▪ Eine Folge von Routes verläuft oft nach diesem Schema:



```

graph LR
  A((Sensor-Knoten)) --> B((Timer-Knoten))
  B --> C((Interpolator))
  C --> D((Ziel-Knoten))
  
```

- Der **Behavior-Graph**:
 - Ergibt sich durch die Menge aller Routes
 - Heißt auch **Route-Graph**, oder **Event-Graph**
 - Ist ein zweiter, dem Szenengraphen **überlagerter** Graph



G. Zachmann Virtuelle Realität und Simulation - WS 09/10 X3D / VRML97 50

Exkurs: Das AEO-Konzept

- In X3D/VRML:
 - actions & objects sind alle Knoten im selben Scenograph
 - Events sind flüchtige "Ereignisse", haben keine greifbare Repräsentation

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 X3D / VRML97 51

Das Execution Model

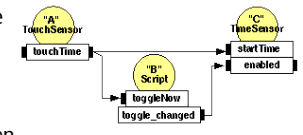
- Die **Event Cascade**:
 - Initialer Event (von Script, Sensor, oder Timer)
 - Propagiere an alle angeschlossenen **eventIn**'s
 - Knoten (z.B. Interpolator) können als Folge weitere Events generieren über **eventOut**'s
 - Alle diese Events sind Teil derselben Kaskade
 - Propagiere so lange, bis die Kaskade leer ist

- Pro Frame können mehrere Kaskaden auftreten (durch verschiedene initiale Events ausgelöst)

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 X3D / VRML97 52

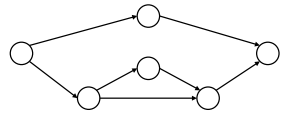
■ Routes induzieren eine Abhängigkeit der Knoten:

- Propagiere in der "richtigen" Reihenfolge
- Algo:
 - Breadth-first traversal
 - Sortiere aktuelle Front gemäß Abhängigkeiten



■ Zyklen:

- Sind erlaubt (manchmal sogar sinnvoll)
- *Loop breaking rule*:
Jedes Feld darf nur 1x pro Event-Kaskade "feuern";
m.A.W.: jede Route wird nur 1x pro Event-Kaskade "bedient"



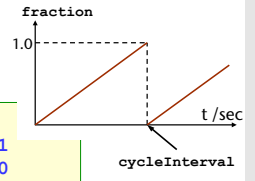
G. Zachmann Virtuelle Realität und Simulation - WS 09/10 X3D / VRML97 53

Knoten für Animationen

■ Der **TimeSensor**-Knoten:

```

TimeSensor {
  exposedField SFTime cycleInterval 1
  exposedField SFTime startTime 0
  exposedField SFBool loop false
  eventOut SFTime fraction_changed
  eventOut SFTime time
  eventOut SFBool isActive
  eventOut SFTime cycleTime
  ...
}
  
```



- Der Timer wird aktiv, sobald die System-Zeit > **startTime** wird
- Falls **startTime** = -1, ist der Timer inaktiv
 - Das ist die beste Art, einen Timer zu de-/aktivieren (s. Bsp. drehende_quadrate)
- Mit Hilfe von **isActive** kann man Animationen aneinanderketten
 - **isActive** zeigt an, ob ein Timer gerade läuft
- **cycleTime** sendet einen **SFTime**-Event bei jedem **cycle**-Beginn

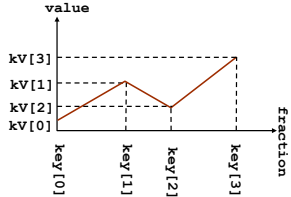
G. Zachmann Virtuelle Realität und Simulation - WS 09/10 X3D / VRML97 54

▪ Ein Knoten zur Interpolation von Skalaren:

```

ScalarInterpolator {
  MFFloat exposedField key
  MFFloat exposedField keyValue
  SFFloat eventIn set_fraction
  SFFloat eventOut value_changed
}

```



▪ Weitere Interpolationsknoten:

```

ColorInterpolator {
  MFVec3f expF keyValue
  SFColor out value_changed
}

CoordinateInterpolator {
  MFVec3f expF keyValue
  MFVec3f out value_changed
}

PositionInterpolator {
  MFVec3f expF keyValue
  SFVec3f out value_changed
}

OrientationInterpolator {
  MFRotation expF keyValue
  SFRotation out value_changed
}

```

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 X3D / VRML97 55

▪ Achtung: man sollte darauf achten, daß man den richtigen Interpolator zum "richtigen" Knoten verbindet

- Es ist z.B. nicht erlaubt, einen **ColorInterpolator** mit der **translation** eines **Transform**-Knotens zu verbinden
- Der **CoordinateInterpolator** ist dazu gedacht, Geometrie zu animieren (also animierte Deformation)
 - Achtung:

$$\text{Anzahl Vec3f's im Feld value_changed} = \frac{\text{Anzahl Vec3f's im Feld keyValue's}}{\text{Anzahl key's}}$$
 und diese Anzahl muß natürlich mit dem Empfänger-Feld übereinstimmen

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 X3D / VRML97 56

Beispiele

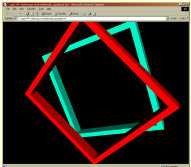
```

DEF Frame1 Transform {
  translation 0.0 0.0 -0.5
  children [ Shape { ... } ]
}
DEF Frame2 Transform {
  translation 0.0 0.0 +0.5
  children [ Shape { ... } ]
}

DEF Rot1 OrientationInterpolator {
  key [ 0.0, 0.5, 1.0 ]
  keyValue [ 0.0 0.0 1.0 0.0, 0.0 0.0 1.0 3.14, 0.0 0.0 1.0 6.28 ]
}
DEF Rot2 OrientationInterpolator {
  key [ 0.0, 0.5, 1.0 ]
  keyValue [ 0.0 0.0 1.0 0.0, 0.0 0.0 1.0 3.14, 0.0 0.0 1.0 6.28 ]
}

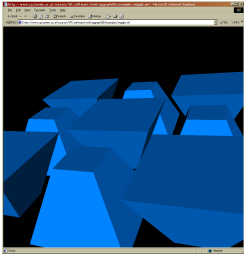
DEF Timer1 TimeSensor { cycleInterval 10.0 loop TRUE startTime -1 }
DEF Timer2 TimeSensor { cycleInterval 11.0 loop TRUE startTime -1 }

ROUTE Timer1.fraction_changed TO Rot1.set_fraction
ROUTE Timer2.fraction_changed TO Rot2.set_fraction
ROUTE Rot1.value_changed TO Frame1.set_rotation
ROUTE Rot2.value_changed TO Frame2.set_rotation
    
```



[drehende_quadrate.wrl](#)

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 X3D / VRML97 57

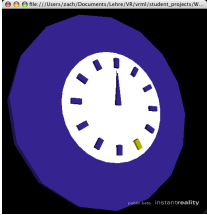


[wiggle.wrl](#)

Beispiel für
CoordinatorInterpolator

```

children USE Wig
Transform ( translation 3.0 0.0 0.0 rotation 1.0 0.0 0.0 3.1415
  children USE Wig )
Transform ( translation 0.0 0.0 -3.0 rotation 1.0 0.0 0.0 3.1415
  children USE Wig )
Transform ( translation -3.0 0.0 -3.0
  children USE Wig )
Transform ( translation 3.0 0.0 -3.0
  children USE Wig )
Transform ( translation 0.0 0.0 3.0 rotation 1.0 0.0 0.0 3.1415
  children USE Wig )
Transform ( translation 1.0 0.0 3.0
  children USE Wig )
Transform ( translation 3.0 0.0 3.0
  children USE Wig )
    
```



[vrml/student_projects/WallClock.wrl](#)

Beispiel für
CoordinatorInterpolator

```

DEF Clock TimeSensor {
  cycleInterval 2.0
  loop TRUE
  startTime 1.0
  stopTime 0.0
}
DEF Interpolator CoordinateInterpolator {
  key [ 0.0, 0.25, 0.5, 0.75, 1.0 ]
  keyValue [
    # 1st coordinate set
    -1.0 1.0 1.0,
    1.0 1.0 1.0,
    1.0 1.0 -1.0,
    -1.0 1.0 -1.0,
    1.0 -1.0 1.0,
    1.0 -1.0 -1.0,
    -1.0 -1.0 1.0,
    -1.0 -1.0 -1.0,
    # 2nd coordinate set
    0.5 1.0 0.5,
    0.5 1.0 -0.5,
    -0.5 1.0 -0.5,
    -0.5 1.0 0.5,
    1.5 -1.0 1.5,
    1.5 -1.0 -1.5,
    -1.5 -1.0 1.5,
    -1.5 -1.0 -1.5,
    # 3rd coordinate set
    -1.0 1.0 1.0,
    1.0 1.0 1.0,
    -1.0 1.0 -1.0,
    1.0 1.0 -1.0,
    -1.0 -1.0 1.0,
    1.0 -1.0 1.0,
    -1.0 -1.0 -1.0,
    1.0 -1.0 -1.0,
    # 4th coordinate set
    ]
}
    
```

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 X3D / VRML97 58

Knoten für User-Input

- Zur Abfrage von User-Input gibt es eine ganze Reihe von **Sensor-Knoten**
- Die meisten haben folgende Felder (zusätzlich zu ihren spezifischen):


```
SFString exposedField description ""
SFBool   eventOut   isOver
SFBool   eventOut   isActive
SFBool   exposedField enabled true
```
- Mit **enabled** kann man einen Sensor de-/aktivieren
- Das Feld **description** is freiwillig, sollte aber unbedingt ausgefüllt werden (sonst weiß man 2 Wochen später überhaupt nicht mehr, welche Funktion ein Sensor hat!)
- isActive=true** gdw. alle Vorbedingungen für einen bestimmten Sensor sind erfüllt
- isOver=true** gdw. die Maus sich über dem "heißen" Bereich des Sensors befindet

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 X3D / VRML97 59

Aktivierung:

- alle Sensor-Knoten sind einem Teil des Szenengraphen zugeordnet
- der Sensor ist aktiv gdw. der User irgend eine Geometrie in diesem Teil (die *sensed geometry*) angeklickt hat (bzw. hält)

Mapping:

- Die meisten Sensor-Knoten mappen die 2D-Maus-Pos. auf eine 3D-Position in der VE (was natürlich nicht notw. die Hauptaufgabe ist)
- Dieses Mapping geschieht durch Schnitt des Strahls vom Viewpoint durch die Maus mit der Szene
- Diesen Schittpunkt kann man mit den Feldern **offset_changed**, **autoOffset**, und **trackPoint_changed** abfragen
 - Diese Felder ex. in allen Sensorknoten; die Bedeutung kann vom konkreten Mapping abhängen

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 X3D / VRML97 60

- **TouchSensor:**
 - erzeugt Ausgaben, sobald die Geometrie geklickt wird
 - wird oft zum Auslösen von Timern verwendet
 - Bsp.: ein virtueller Button soll eine Tür öffnen
- **PlaneSensor:**
 - konvertiert die select-and-drag-Bewegung in eine 3D-Bewegung, die aber auf eine Ebene im Raum eingeschränkt ist
 - diese Ebene ist die lokale $z=0$ Ebene
 - wird oft zum Bewegen von Geometrie auf einer Ebene verwendet
 - Verbinde das Feld `translation_changed` mit `set_translation` eines `Transform`'s
 - Die Koordinaten von `translation_changed` sind **relativ zum lokalen Koord.system!**

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 X3D / VRML97 61

- **CylinderSensor:**
 - Funktioniert ähnlich wie der PlaneSensor
 - Unterschied: die Maus-Position wird auf einen Zylinder gemapt
 - Ausgabe-Feld: `rotation_changed`
 - Häufiger Verwendung: zur Implementierung von Drehknöpfen
 - Gibt sehr viele weitere Parameter-Felder
- **SphereSensor:**
 - you get the idea by now ...

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 X3D / VRML97 62

- **KeySensor:**
 - Interface zum Keyboard
 - Ausgabe-Felder:


```
SFInt32  actionKeyPress
SFInt32  actionKeyRelease
SFString keyPress
SFString keyRelease
```
- **StringSensor:**
 - String-basiertes Interface zum Keyboard
 - Erlaubt die Eingabe kompletter Strings

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 X3D / VRML97 63

Utility-Knoten für Animationen

- Diese sind soz. "glue nodes"
 - Z.B.: SFBool → SFTime, SFFloat → SFVec3f
- Sequencer-Knoten:
 - Diskrete Variante der Interpolator-Knoten
 - Ausgabe-Feld **value_changed** nimmt nur Werte aus dem Feld **keyValue** an
 - Keine Interpolation
 - Das Ausgabe-Feld ändert seinen Wert nur, wenn der Wert im Feld **set_fraction** von einem Intervall [**key[i]**, **key[i+1]**) in ein anderes Intervall [**key[j]**, **key[j+1]**) wechselt
 - Varianten: **BooleanSequencer**, **IntegerSequencer**, **FloatSequencer(?)**

BooleanSequencer
value_changed

IntegerSequencer
value_changed

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 X3D / VRML97 64

- **Trigger-Knoten**
 - **BooleanTrigger:** konvertiert SFTIME → SFBool
 - Erzeugt immer einen True-Event, wenn ein Event eingeht
 - **IntegerTrigger:** konvertiert SFBool → SFInt32
 - Der ausgegebene Wert kann an einem weiteren Feld spezifiziert werden
 - **TimeTrigger:** konvertiert SFBool → SFTIME
 - Erzeugt die aktuelle System-Zeit am Ausgabe-Feld, wenn ein Event eingeht

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 X3D / VRML97 65

- **BooleanFilter:**

```
SFBool eventIn    set_boolean
SFBool eventOut  inputFalse
SFBool eventOut  inputNegate
```

 - **inputFalse** liefert True-Event, wenn **set_boolean** False-Event empfängt
 - **inputNegate** liefert den negierten Event von **set_boolean**
- **BooleanToggle:**
 - "Toggle switch" = Kippschalter

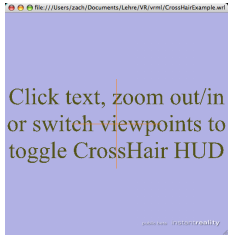

```
SFBool eventIn    set_boolean
SFBool exposedField toggle
```

 - Das Feld **toggle** kippt jedesmal, wenn ein True-Event empfangen wird

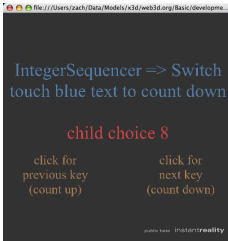
G. Zachmann Virtuelle Realität und Simulation - WS 09/10 X3D / VRML97 66

Beispiele

- Beispiel für
 - Geometrie, die fest bleibt relativ zum Viewpoint
 - **Switch** (und **PROTO**)
 - **ProximitySensor**, **TouchSensor**,
 - **BooleanToggle**, **-Trigger**, **-Filter**
- Beispiel für
 - **IntegerSequencer** und **Switch**



CrossHair.wrl



IntegerSequencer.wrl

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 X3D / VRML97 67

Script-Knoten

- Erlauben die Implementierung von komplexen Verhalten in einer sog. "Skript"-Sprache
 - Der Standard läßt die konkrete Sprache offen; er definiert statt dessen ein sog. *Scene Access Interface (SAI)*
 - Früher *EAI (External Access Interface)*
 - Typisch sind Java und/oder Javascript (aka ECMAScript)
 - Für diese gibt es dann sog. Bindings, d.h., ein konkretes API, das das SAI impl.
 - InstantReality kann beides, FreeWRL ?
- IMHO ein Design-Bug im Standard: "Browsers are not required to support any specific language"

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 X3D / VRML97 68

- Deklaration / Syntax:

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

- Zur Erinnerung die Access Types:

VRML97	Bedeutung
eventIn	nur Eingabe
eventOut	nur Ausgabe
field	nur Daten
exposedField	Kombination

- Das **url**-Feld:

- Link auf einen Javascript-File:

```
url [ "MyBehavior.js"
      "http://www.my.site/MyBehavior.js" ]
```

- Link auf einen Java-File:

```
url [ "MyBehavior.class"
      "http://www.my.site/MyBehavior.class" ]
```

- Javascript-Source-Code:

```
url [ "javascript:
      var x, y, z;
      function initialize( timeStamp )
      {
        ...
      }
      " ]
```

Zugriff auf die Felder

- Zu jedem **eventIn**-Feld gibt es eine Funktion mit demselben Namen
 - Heißt **Event-Handler**
 - Wird irgendwann innerhalb der Event-Kaskade aufgerufen, falls Event eintrifft
 - Parameter: Wert des Events (Kopie der Daten) & Timestamp
- Zu jedem **eventOut**-Feld gibt es eine implizit vordefinierte Variable mit dem Namen **xxx_changed** und passendem Typ
 - Schreiben der Variable = Generieren eines Events
 - Event wird nur 1x pro Time-Stamp erzeugt!
- Zu jedem **eventIn**-Feld gibt es eine implizit vordefinierte Variable mit dem Namen **set_xxx**

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 X3D / VRML97 71

Beispiel

```

DEF SomeNode Transform
{
  translation 0 0 0
  children [ ... ]
}

Script
{
  field SFNode tnode USE SomeNode
  eventIn SFVec3f pos
  directOutput TRUE
  url [ "javascript:
      function pos(value, timestamp)
      {
        tnode.set_translation = value;
      }
    " ]
}

```

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 X3D / VRML97 72

Bemerkungen

- Das Feld `directOutput` muß man auf `TRUE` setzen, falls der Script-Knoten andere Knoten direkt manipuliert; falls `directOutput= FALSE`, dann darf der Script-Knoten den Rest der Szene nur über Routes modifizieren!
- Innerhalb einer Event-Kaskade bekommen alle Funktionsaufrufe denselben Timestamp
- Innerhalb eines Frames können die Event-Handler mehrfach aufgerufen werden
- Spezielle Funktionen (hier in Javascript):
 - `eventsProcessed()` : wird am Ende einer Event-Kaskade aufgerufen
 - `prepareEvents()` : wird vor dem Route-Processing aufgerufen
 - Z.B. zur Abfrage von externen Geräten
 - `initialize()` : wird direkt nach dem Laden der Szene aufgerufen

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 X3D / VRML97 73

- Alle Knoten haben 2 (bisher verschwiegene) Felder

```
inputOnly MFNode addChildren
inputOnly MFNode removeChildren
```

- Das Execution-Model mit Skript-Knoten:

```
1. Update Camera (based on currently active Viewpoint node)
2. Evaluate Sensor nodes and
   all script-nodes' prepareEvents() function
   (→ initial events)
3. for all initial events:
4.   process the event cascade, i.e., route events
5.   if any script node was visited during routing:
6.     evaluate its eventsProcessed() function
7. Render scene; swap rendering buffers
8. Browser's clock ← system clock (one "clock tick")
```

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 X3D / VRML97 74

- Output-Events eines Skripts bekommen denselben Timestamp wie dessen Input-Events
- Zugriff auf Elemente eines Vektors (z.B. **SFVec3f**):
 $v[0]$, $v[1]$, $v[2]$
- Zugriff auf Elemente eines **MF**-Feldes (Arrays): **values [0]**, ...
 - Beispiel: **MFRotation values** → **values [0] [3]** = Winkel der 1. Rot

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 X3D / VRML97 75

Beispiele

- Zur Laufzeit generierte Geometrie:

<examples/CreateVrmlFromStringRandomBoxes.wrl>

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 X3D / VRML97 76

- Beispiel für
 - "printf"-Ausgabe auf der Konsole des Browsers
 - Javascript-Knoten
 - Zugriff auf andere Knoten der Szene mittels

```
field SFNode AliasNode USE SceneNode
```

- Billboard
- Funktionalität: schaltet verschiedene Viewpoints durch



<examples/ViewpointSequencer.wrl>

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 X3D / VRML97 77

Ein Bug in der X3D-Spezifikation

- Felder mit einem Namen zzz, die den Access-Type exposedField haben, haben implizit den Namen zzz_changed, wenn sie als Ausgabe-Feld verwendet werden, und den Namen set_zzz, wenn sie als Eingabe-Feld verwendet werden
- Problem mit inputOutput-Feldern in Script-Knoten:
 - Es wird ein Event-Handler (= Funktion) zzz() definiert ...
 - ... und eine Variable zzz
 - Ist in Javascript nicht erlaubt, da Funktionen und Variablen im selben Namespace leben!

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 X3D / VRML97 78

Prototypes

- Definition neuer Knotenarten; faßt zusammen:
 - Knoten (Shapes, Sensors, Interpolators, etc.)
 - Script-Knoten
 - Routes
- Beispiel:


```

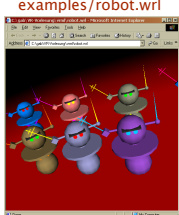
                PROTO Robot
                [ field SFCOLOR eyeColor 1.0 0.0 0.0
                  ...
                ]{
                  Shape { appearance Appearance {
                            material Material {
                                diffuseColor IS eyeColor
                            }
                        }
                  ...
                }
            
```

Name der neuen Knoten-Klasse

Zuweisung des Interfaces

Interface (Felder & Events)

Body (Implementierung)




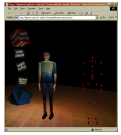
examples/robot.wrl

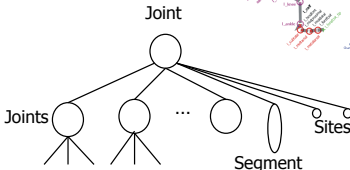
G. Zachmann Virtuelle Realität und Simulation - WS 09/10
X3D / VRML97 79

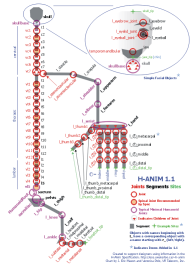
Weiterentwicklungen

- In X3D:
 - Es kommen laufend neue Knoten hinzu, z.B. für Shader
- H-Anim:
 - Standard zur Spezifizierung von "humanoiden" Figuren in VRML97/X3D
 - Joints = Baumstruktur
 - Segments = Geometrie
 - Sites = "Handles" (ausgezeichnete Punkte im Koord.system des Joints)









G. Zachmann Virtuelle Realität und Simulation - WS 09/10
X3D / VRML97 80

Tips & Tricks zum Entwickeln mit X3D/VRML

- Editor:
 - Man wird früher oder später X3D-Code "von Hand" editieren müssen
 - ASCII-Editor verwenden, der Syntax-Highlighting für VRML/X3D hat!
 - ... und der Klammer-Matching beherrscht!
- X3D-Edit ?

G. Zachmann Virtuelle Realität und Simulation - WS 09/10 X3D / VRML97 81

Debugging

- Im wesentlichen nur "printf"-Debugging möglich
- Beispiel: ein Debug-Knoten

```

DEF Debug Script {
  eventIn MFVec3f set_coord
  eventIn SFFloat set_float
  url [ "javascript:
    function set_coord( value, timestamp )
    {
      print( 'Debug: coord = ' + value + '\n' );
    }
    function set_float( value, timestamp )
    {
      print( 'Debug: float = ' + value + '\n' );
    }
  " ]
}
```

Beispiel in [examples/wiggle_with_debug.wrl](#)


G. Zachmann Virtuelle Realität und Simulation - WS 09/10 X3D / VRML97 82

- Alternative in InstantReality:
 - Mittels des Nicht-Standard-Knotens **Logger**

```
DEF Log Logger
{
  level 3      # 0 - ..
  logFile ""   # default = console
}
ROUTE Clock.fraction_changed TO Log.write
```

X3D / VRML97 83

- Extrem praktisches Feature: Der X3D-Browser von InstantReality erlaubt es, zur Laufzeit den Szenengraphen zu beobachten und sogar Felder zu verändern!
- Anleitung:
 - In InstantPlayer: Help → Web Interface Scenegraph
 - Ein Browser-Fenster öffnet sich
 - In der Tabelle klicken: "Named" → "scene (Scene)"
- Demo:
 - Szene: eg2001.competition/ah2k/ah2k.wrl
 - Links: Named → scene (Scene) → DEF spinClock TimeSensor → enabled TRUE/FALSE




System:

User: zach
Host: r02achneu.in.tu-berlin.de
HostID: 952a2a7f9194e91244d2310184
Server: http://r02achneu.in.tu-berlin.de:35668
Version: V2.0.0beta2 build: R-8787 Sep 13 2007
Content: file:///C:/Users/zach/Desktop/InstantVR/

Generic Pages:

Object Type	Count	Page
Scene Object	1514	
Route	87	
SceneNode	86	All
Node	108	Named
NameSpace	4	All
Node Type	153	All
Field Type	51	All
Photo	0	

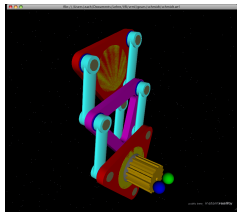


AirHockey 2000
Click The Table to Begin


X3D / VRML97 84

Demos

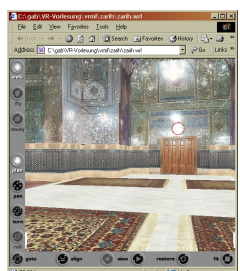
Veranschaulichung von komplizierten Kinematiken (hier: *Schmidt Offset Coupling*)



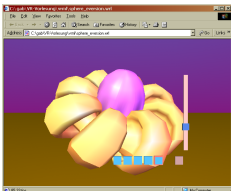
AirHockey 2000
Click The Table to Begin



Spiele
(Quelle: Eurographics 2001 competition)



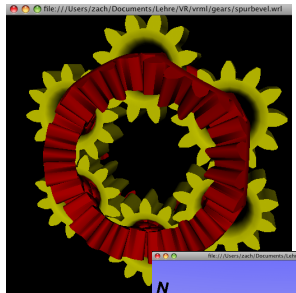
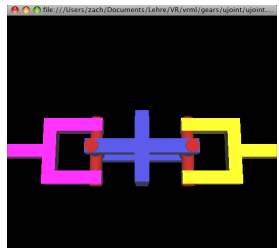
Cultural heritage
(Quelle: www.aqrazavi.org)

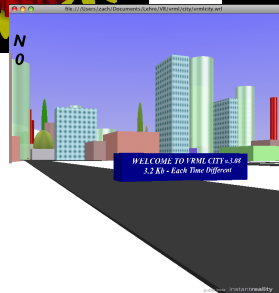
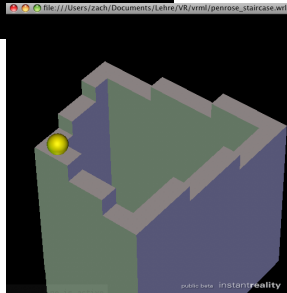


Edutainment (Wissenschaft?), Wissenstransfer
Bsp.: *sphere eversion*

G. Zachmann Virtuelle Realität und Simulation - WS 09/10

X3D / VRML97 85

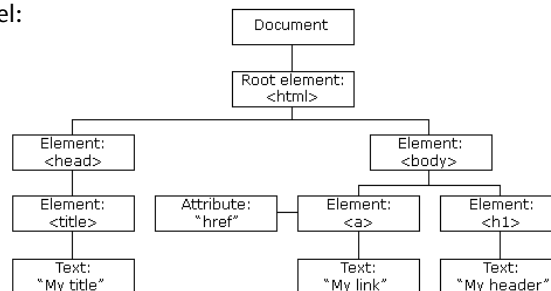



G. Zachmann Virtuelle Realität und Simulation - WS 09/10

X3D / VRML97 86

Ausblick: X3DOM

- Aktuelle Bestrebung des W3C, X3D als festen Bestandteil von HTML5 zu integrieren
- DOM = Document Object Model = Spezifikation ...
 1. wie ein HTML/XML-Dokument als Baum repräsentiert wird; und
 2. wie man auf die Elemente des Baumes zugreift
- Beispiel:



G. Zachmann Virtuelle Realität und Simulation - WS 09/10

X3D / VRML97 87

Ein mögliches Beispiel

```

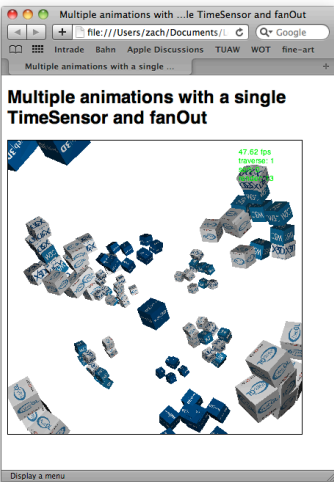
...
<body>
  <h1>X3D DOM Events</h1>
  <x3d xmlns="http://www.web3d.org/specifications/x3d-3.0.xsd">
    <Scene>
      <Transform>
        <Shape>
          <Box size="4 4 4" />
        </Shape>
        <TouchSensor id="ts" DEF="ts" />
      </Transform>
    </Scene>
  </x3d>
  <script type="text/javascript">
    // The namespace URIs
    var xhtml_ns = "http://www.w3.org/1999/xhtml";
    var x3d_ns = "http://www.web3d.org/specifications/x3d-3.0.xsd";
    // Get elements using namespaces
    var h1=document.getElementsByTagNameNS( xhtml_ns, "h1" );
    var x3d=document.getElementsByTagNameNS( x3d_ns, "x3d" )[0];
    var ts = x3d.getElementsByTagName( "TouchSensor" )[0];
    ts.addEventListener( "touchTime",
      function() {alert("clicked"); },
      false );
  </script>
</body>

```

G. Zachmann Virtuelle Realität und Simulation - WS 09/10

X3D / VRML97 88

- Demo:
demos/x3dom/fanOutRoute.xhtml
- Weiterführender Link:
<http://www.x3dom.org/>
 - Beta-Versionen, Doku, FAQs,



G. Zachmann Virtuelle Realität und Simulation - WS 09/10 X3D / VRML97 89



G. Zachmann Virtuelle Realität und Simulation - WS 09/10 X3D / VRML97 90