


11. Computergrafik

- 11.1 Grundlagen der 2D-Computergrafik
- 11.2 2D-Vektorgrafik mit XML: SVG
- 11.3 Grundlagen der 3D-Computergrafik
- 11.4 3D-Computergrafik: VRML (Fortsetzung) 

NavigationInfo

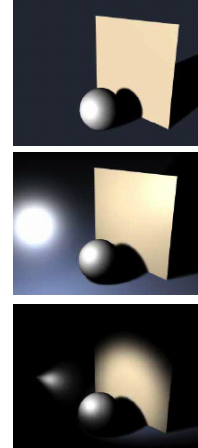
- Gibt globale Zusatzinformation für das Rendering an:
 - Z.B. Standardmodus (Walk, Fly, Examine)
 - Z.B. Standardgeschwindigkeit
- Headlight:
 - Standardlichtquelle (directional) aus Betrachtersicht
 - kann in NavigationInfo ausgeschaltet werden

- Beispiel:

```
NavigationInfo {  
  type "EXAMINE"  
  headlight FALSE  
}
```

Szenenbeleuchtung: Lichttypen

- Drei Lichttypen werden in VRML unterstützt (eigene Knotentypen)
- **Directional Light:**
 - parallel gerichtetes Licht einer unendlich weit entfernten Quelle
 - keine Abschwächung mit der Entfernung
- **PointLight:**
 - Licht breitet sich gleichmäßig von punktförmiger Quelle aus (z.B. Glühlampe)
 - Abschwächung mit der Entfernung
- **SpotLight:**
 - Licht breitet sich kegelförmig von punktförmiger Quelle aus (z.B. Taschenlampe)
 - Abschwächung mit der Entfernung und mit Winkel
- Wichtigste Feldtypen:
 - **direction**-Feld: Richtungsvektor
 - **ambientIntensity**-Feld: Stärke des Einflusses auf Objekte
 - **color**-Feld: Lichtfarbe
 - **location**-Feld: Position der Lichtquelle



Beispiel: Szene mit Beleuchtung

```
DirectionalLight {
  direction 0 -1.0 0
  ambientIntensity 0.7
  color 1.0 1.0 1.0
}

SpotLight {
  location -5.0 3.0 0
  direction 0.5 -0.5 0.0
  ambientIntensity 0.4
  color 1.0 1.0 1.0
}

Group {
  ... wie früheres Beispiel zu Gruppen
}

NavigationInfo {
  headlight FALSE
}
```

PositionInterpolator

- Zweck:
 - Bewegung von Objekten in VRML-Animationen
- Schlüsselwerte:
 - Entsprechend der Konventionen von **translation**-Feldern in **Transform**-Knoten
 - D.h.:
 - » 3 Werte für aktuelle Position

```
– Beispiel:  
  keyValue [  
    0 3 0,  
    0 0.5 0,  
    0 1.5 0,  
    0 2 0,  
    0 1.5 0,  
    0 0.5 0,  
    0 1 0,  
    0 0.5 0,  
    0 3 0  
  ]
```

Beispiel: Fallende Kugel

```
DEF Ball Transform {  
  children [  
    Shape {  
      appearance Appearance {  
        material Material {  
          diffuseColor 0.5 0 0.8  
        }  
      }  
      geometry Sphere {  
        radius 0.5  
      }  
    ]  
  }  
}  
  
DEF Plane Shape {  
  appearance Appearance {  
    material Material {  
      diffuseColor 0 1 0  
    }  
  }  
  geometry Box {  
    size 4 0.1 4  
  }  
}  
  
DEF Clock TimeSensor {  
  cycleInterval 6.0  
  loop TRUE  
}  
  
DEF Interpolator  
  PositionInterpolator {  
    key [0.0, 0.05, 0.1, 0.15,  
        0.2, 0.3, 0.4, 0.5, 1.0]  
    keyValue [  
      0 3 0,  
      0 0.5 0,  
      0 1.5 0,  
      0 2 0,  
      0 1.5 0,  
      0 0.5 0,  
      0 1 0,  
      0 0.5 0,  
      0 3 0  
    ]  
  }  
}  
  
ROUTE Clock.fraction_changed TO  
  Interpolator.set_fraction  
ROUTE Interpolator.value_changed  
  TO Ball.set_translation
```

ColorInterpolator

- Zweck:
 - Veränderung der Farbe von Objekten in VRML-Animationen
- Schlüsselwerte:
 - Entsprechend der Konventionen zur Darstellung von RGB-Farben
 - » 3 Werte für aktuellen Farbton

– Beispiel:

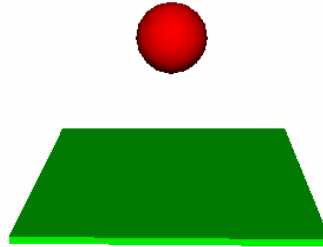
```
keyValue [  
  0.5 0 0.8, #violett  
  0.5 0 0.8,  
  0.5 0 0.8,  
  0.5 0 0.8,  
  0.5 0 0.8,  
  0.5 0 0.8,  
  0.5 0 0.8,  
  0.5 0 0.8,  
  0.5 0 0.8,  
  1 0 0, #rot  
  1 0 0,  
  0.5 0 0.8  
]
```

Beispiel: Kugelverfärbung (1)

```
...  
DEF Ball Transform {  
  children [  
    Shape {  
      appearance Appearance {  
        material DEF Color Material {  
        }  
      }  
      geometry Sphere {  
        radius 0.5  
      }  
    }  
  ]  
}  
...  
DEF PInterpolator PositionInterpolator {  
  ...  
}  
...
```

Beispiel: Kugelverfärbung (2)

```
...
DEF CInterpolator ColorInterpolator {
  key [0.0, 0.05, 0.1, 0.15, 0.2, 0.3,
       0.4, 0.5, 0.55, 0.95, 1.0]
  keyValue [
    0.5 0 0.8,
    0.5 0 0.8,
    0.5 0 0.8,
    0.5 0 0.8,
    0.5 0 0.8,
    0.5 0 0.8,
    0.5 0 0.8,
    0.5 0 0.8,
    1 0 0,
    1 0 0,
    0.5 0 0.8
  ]
}
ROUTE Clock.fraction_changed TO PInterpolator.set_fraction
ROUTE PInterpolator.value_changed TO Ball.set_translation
ROUTE Clock.fraction_changed TO CInterpolator.set_fraction
ROUTE CInterpolator.value_changed TO Color.set_diffuseColor
```



ScalarInterpolator

- Universeller Interpolator
- Berechnet beliebige reelle Zahlenwerte abhängig von gegebenen Schlüsselwerten
- Anwendungsbeispiele:
 - Animation der Grösse von Objekten
 - Animation der Transparenz von Objekten

Sensoren für Interaktionen in VRML

- TimeSensor: Uhr, Zeitgeber
- TouchSensor: Berührung mit dem Mauszeiger
- PlaneSensor: Verschiebung von Objekten in einer Ebene
- SphereSensor: Freie Rotation eines Objektes
- CylinderSensor: Rotation eines Objektes um eine Achse
- ProximitySensor: Nähe des Beobachters
- Collision: Kollision eines Beobachters mit dem Objekt
- LOD (=Level of Detail): Entfernung des Beobachters zum Objekt
- Anchor: Hyperlink

TouchSensor-Knoten

- Knotentyp **TouchSensor**
 - Kann an verschiedenen Stellen in Objekthierarchie eingebaut werden
- Ereignis (**EventOut**) **isOver**:
 - Erzeugt Ereignis, wenn sich Mauszeiger im Objekt befindet
- Ereignis (**EventOut**) **isActive**:
 - Erzeugt Ereignis, wenn Maus im Objekt gedrückt ist

Beispiel: Interaktion in Ball-Animation

```
DEF Ball Transform {
  ...
}

Group {
  children [
    DEF Sensor TouchSensor {}
    DEF Plane Shape {
      ...
    }
  ]
}

DEF Clock TimeSensor {...}
DEF PInterpolator PositionInterpolator {...}
DEF CInterpolator ColorInterpolator {...}

ROUTE ...
ROUTE Sensor.isOver TO Clock.set_enabled
```

Beispiel: Kollisionserkennung

```
Collision {
  children [
    Transform {
      children [
        Shape {
          appearance Appearance {
            material Material {
              diffuseColor 1 0 0
            }
          }
          geometry Sphere {
            radius 0.5
          }
        }
      ]
    }
  ]
  translation 0 0 -5
}
collide TRUE # Verhindert Eindringen in Objekt
}
```

Benutzerdefinierte Formen

- Beliebige Formen können über Koordinatenwerte definiert werden
 - Knotentyp `Coordinate`, Feldtyp `point`, Werte 3er-Gruppen von reellen Zahlen
- Bildung von Objekten mit `IndexedLineSet` bzw. `IndexedFaceSet`:
 - `IndexedLineSet` erzeugt Gittermodell, `IndexedFaceSet` Flächenmodell
 - Feld `coord` enthält die beteiligten Punkte
 - » Implizit werden die Punkte, mit 0 beginnend, nummeriert (je drei Zahlen = 1 Punkt)
 - Feld `coordIndex` enthält die einzelnen anzuzeigenden Linien bzw. Flächen
 - » Als Indizes in der Punktliste
 - » Jedes Element (Linie bzw. Fläche) mit -1 abgeschlossen

Beispiel: Würfel selbstdefiniert (Drahtgitter)

```
shape {
  appearance ...
  geometry IndexedLineSet {
    coord Coordinate {
      point [
        -1.0 1.0 1.0, # Punkt 0: links oben vorn
        -1.0 -1.0 1.0, # Punkt 1: links unten vorn
        1.0 -1.0 1.0, # usw.
        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
      ]
    }
    coordIndex [
      0, 1, 2, 3, 0, -1, # vorderes Quadrat
      4, 5, 6, 7, 4, -1,
      0, 4, -1,
      1, 5, -1,
      2, 6, -1,
      3, 7
    ]
  }
}
```


Beispiel: Würfel selbstdefiniert (Flächen)

```
Shape {
  appearance ..
  geometry IndexedFaceSet {
    solid FALSE
    coord Coordinate {
      point [... wie oben ...]
    }
    coordIndex [
      0, 1, 2, 3, 0, -1,
      4, 5, 6, 7, 4, -1,
      0, 3, 7, 4, 0, -1,
      1, 2, 6, 5, 1, -1,
      3, 2, 6, 7, 3, -1,
      0, 1, 5, 4, 0
    ]
  }
}
```

Programmeinbindung in VRML: Script-Knoten

- Der Knotentyp `script` ermöglicht die Einbindung eines Programmskripts
 - Meist JavaScript, VRMLScript (spezialisierte Teilsprache von JavaScript) oder Java
- Skript kann (nach entsprechender Deklaration)
 - Feldwerte lesen und verändern
 - Eingabeereignisse verarbeiten
 - Ausgabeereignisse erzeugen

External Authoring Interface

- Ermöglicht Datenaustausch zwischen einer VRML-Welt und einem Java-Applet
 - Voraussetzung: Applet und VRML-Welt in gleiche Webseite integriert
- Beispiel einer Anwendung:
 - Komplexe Visualisierung, z.B. von geographischen Daten, in mehreren Fenstern
 - Fenster 1: Dreidimensionale Ansicht (VRML-Welt)
 - Fenster 2: Navigations- oder Übersichtsfenster, z.B. Landkarte mit aktuellem Ausschnitt oder Standort
- Ermessensentscheidung beim Programmentwurf:
 - VRML-Dokument mit hohem Programmlogik-Anteil oder
 - Programm mit Einbindung von VRML-Dokumenten (z.B. Java 3D)

Aktuell: X3D

- Neuer Standard (Web3D-Konsortium):
 - X3D (eXtensible3D-Graphics)
- XML-Anwendung
 - Dateien folgen XML-Syntax (alternativ VRML-Encoding)
- Enge Integration mit anderen Web-Technologien
- Allgemeine XML-Werkzeuge anwendbar
 - Z.B. Navigation mit XPath
 - Z.B. Transformation mit XSLT
- Entwicklungsstand:
 - Sommer 2003: Draft International Standard ISO 19775
 - Industrielle Unterstützung derzeit allerdings vorwiegend durch kleine Nischenfirmen (z.B. „Media Machines“, Tony Parisi)
- Übergang von VRML zu X3D:
 - X3D-Viewer geben im Allgemeinen auch VRML wieder
 - Transformationsprogramme

Neue Features in X3D gegenüber VRML

- Verbessertes Rendering
- Non-Uniform Rational B-Spline Surfaces (NURBS)
 - Standard zur mathematischen Modellierung gewölbter Flächen
 - Weit verbreitet in 3D-Modellierungswerkzeugen
 - Proprietäre Erweiterungen für VRML existieren ebenfalls
- Geospatiale Referenzen
- Humanoide Animation
- Distributed Interactive Simulation (DIS)
 - vorwiegend militärisches Interesse

Trends in 3D-Auszeichnungssprachen

- Streaming
 - Laden der virtuellen Welt vom Server nach Bedarf
 - Kompression von 3D-Daten
 - Integration von Realzeit-Medien (z.B. Kamerabilder)
- Realzeit-Interaktion mit 3D-Welten
- Integration mit anderen Medien
 - Universelle „Player“ für Filme, Musik, Animationen, 3D-Welten, ...
 - Z.B. Integration von Szenegraphen in MPEG-4 (XML-basierte Sprache XMT)
 - z.B. MediaMachines Flux Player