

10. Computergrafik

- 10.1 Grundlagen der 2D-Computergrafik
- 10.2 2D-Vektorgrafik mit XML: SVG
- 10.3 Grundlagen der 3D-Computergrafik
- 10.4 3D-Computergrafik: VRML (Fortsetzung) 

Animation von 3D-Grafik

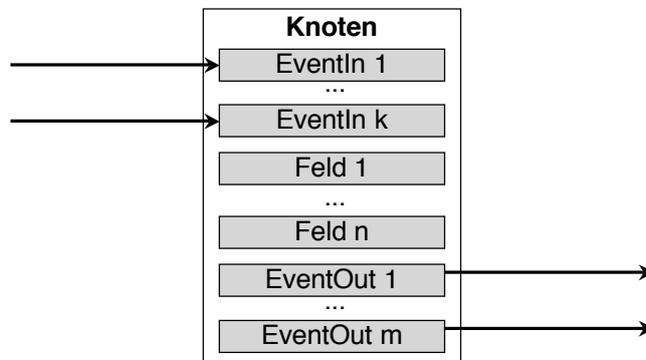
- *Dynamik* in der Darstellung von 3D-Szenen - drei mögliche Ursachen:
 - Veränderung der Betrachterposition
 - » auch in *statischen* 3D-Szenen möglich
 - Automatische, d.h. von selbst ablaufende, Veränderungen innerhalb der 3D-Szene
 - » z.B. Bewegung oder Veränderung von Farbe/Beleuchtung
 - » *Animation*
 - Vom Benutzer oder anderen systemexternen Quellen gesteuerte Veränderungen innerhalb der 3D-Szene
 - » *Interaktion*
- Wesentlich sowohl für Animation als auch Interaktion:
 - Veränderung des Modells als Reaktion auf *Ereignisse* (*dynamische* Szenen)
 - » Zeitereignisse und externe Ereignisse
 - » Ausschliesslich Zeitereignisse und deren Folgeereignisse = Animation

Ereignisse in VRML

- Ereignisentstehung:
 - *Sensoren*, eine spezielle Art von VRML-Objekten, erzeugen Ereignisse zu bestimmten Zeitpunkten
- Für Animation wichtig:
 - Zeitsensoren (Taktgeber)
- Für Interaktion wichtig:
 - Sensoren für Benutzerinteraktion (z.B. TouchSensor)
- Ereignisverarbeitung:
 - Ereignisse können an beliebige Objekte weitergeleitet (*routed*) werden
 - In einem empfangenden Objekt
 - » können Veränderungen von Attributen ausgelöst werden
 - » können erneut Ereignisse ausgesendet werden (*Ereigniskaskade*)
- Spezialobjekte zur Ereignisumsetzung:
 - insbesondere *Interpolatoren* zur drastischen Reduzierung der zu betrachtenden Ereignisanzahl

Knotenattribute in VRML

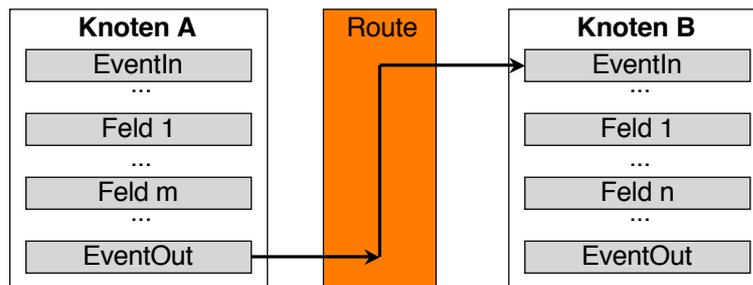
- Knoten können drei Arten von Attributen haben:
 - Felder zur statischen Festlegung von Eigenschaften
 - *EventIn*-Attribute zum Empfang von Ereignissen
 - *EventOut*-Attribute zum Senden von Ereignissen



Ereignisweitergabe auf Routen

- Ereignisse werden an spezifische Zielobjekte weitergegeben
- Das Zielobjekt wird nicht beim Erzeugen des Ereignisses spezifiziert, sondern in einem speziellen Sprachkonstrukt: *Route*

`ROUTE KnotenA.EventOut TO KnotenB.EventIn`



Fan-Out (mehrere Empfänger für gleiches Ereignis) unproblematisch
Fan-In (mehrere Ereignisse gleichzeitig an ein Objekt) problematisch

Ereignisempfang

- Die meisten Knotentypen unterstützen (**EventIn-**) Ereignisse der Art `set_Feldwert`
- Beispiele für Verwendung:
 - Setzen der absoluten Position (translation) in einem **Transform**-Knoten
 - Setzen von Rotationswerten in einem **Transform**-Knoten
 - Setzen von Farbwerten

Zeitsensoren

- Zeitsensor:
 - Uhr, die regelmässig Zeitereignisse generiert (Taktgeber)
- Zwei Verwendungsarten:
 - Absolute Zeit (normale Uhr):
 - » Gibt verstrichene Zeit seit Referenzzeitpunkt an (1. Januar 1970, 0:00 Uhr GMT)
 - Relative Zeit (Stopp-Uhr): *wesentlich häufiger verwendet!*
 - » Gesamtdauer des Ablaufs festgelegt
 - » Relative Zeit beginnt bei Start mit 0 und überschreitet nie die Gesamtdauer des Ablaufs
 - » Automatische Wiederholung (= Rücksprung der relativen Zeit zu 0) möglich
 - » Relative Zeiten werden in VRML als Bruchteile der Gesamtdauer (Reelle Zahlen zwischen 0 und 1) angegeben

TimeSensor-Knoten

- Knotentyp **TimeSensor**
 - Erlaubt diverse Feldtypen
 - Erzeugt Ereignisse
- Feldtyp **enabled**: Uhr ein/aus
- Feldtyp **startTime**: Startzeit, default 0.0
- Feldtyp **stopTime**: Endzeit, default 0.0
- Feldtyp **cycleInterval**: Zeitintervall für relative Zeitmessung
- Feldtyp **loop**: Wiederholung ein/aus, default **FALSE**
 - Endlosschleife möglich durch **stopTime = startTime**
- Ereignis (**EventOut**) **time**: absolute Zeit
- Ereignis (**EventOut**) **fraction_changed**: relative Zeit
 - wichtig zur Steuerung von Animationen

Beispiele für (relative) Zeitsensoren in VRML

- Einmaliger Ablauf von Dauer 6 Sekunden

```
DEF Clock TimeSensor {
  cycleInterval 6.0
}
```
- Endlosschleife mit Periode 6 Sekunden

```
DEF Clock TimeSensor {
  cycleInterval 6.0
  loop TRUE
}
```
- Vier Durchläufe mit Periode 6 Sekunden, insgesamt 24 Sekunden

```
DEF Clock TimeSensor {
  cycleInterval 6.0
  loop TRUE
  stopTime 24.0
}
```

Interpolatoren

- *Interpolatoren* dienen zur Schlüsselwert-orientierten Definition von Animationen
 - Vollständige Angabe von Animationen zu umfangreich
 - Schlüsselwert: Definierter Wert (*key value*) zu einem bestimmten Zeitpunkt (*key*)
- Interpolator berechnet durch lineare Interpolation alle Zwischenwerte zwischen den gegebenen Schlüssel-/Wert-Paaren
- Typische Anwendung von Interpolatoren in Ereignisverarbeitung:



Genereller Aufbau von Interpolatoren

- Alle Interpolatoren in VRML haben folgende Elemente
- Feldtyp **key**
 - Liste mit Zeitwerten, zu denen Schlüsselwerte festgelegt werden sollen
 - Müssen den Gesamtzeitraum *nicht* linear aufteilen
 - z.B.: **key** [0.0, 0.15, 1.0]
- Feldtyp **keyValue**
 - Liste mit Schlüsselwerteinstellungen für die angegebenen Zeitpunkte
 - Sollten genau das Format aufweisen, das der empfangende Ausführungsknoten erwartet
 - Sinnvoll: Gleich viele Werte wie Zeitwerte im zugehörigen **key**-Feld
 - z.B.: **keyValue** [0 1 0 0.00, 0 1 0 1.57, 0 1 0 3.14]
- Eingabeereignis (EventIn) **set_fraction**
 - Passend zu den Ausgabeereignissen von Zeitsensoren
- Ausgabeereignis (EventOut) **value_changed**
 - Zur Weitergabe von Feldwertänderungen an Ausführungsknoten

OrientationInterpolator

- Zweck:
 - Drehung von Objekten in VRML-Animationen
- Schlüsselwerte:
 - Entsprechend der Konventionen von **rotation**-Feldern in **Transform**-Knoten
 - D.h.:
 - » 3 Werte für Rotationsachse
 - » 1 Wert für Rotationswinkel
 - Beispiel:

```
keyValue [  
  0 1.0 0 0.00,  
  0 1.0 0 1.57,      =  $\pi/2$   
  0 1.0 0 2.36,      =  $3\pi/4$   
  0 1.0 0 3.14  
]
```

Beispiel: Würfeldrehung

```
DEF RotCube Transform {
  children [
    Shape {
      appearance Appearance {
        texture ImageTexture {
          url "textur0.gif"
        }
      }
      geometry Box {
        size 2.0 2.0 2.0
      }
    }
  ]
}

DEF Clock TimeSensor {
  cycleInterval 6.0
  loop TRUE
}

DEF Interpolator OrientationInterpolator {
  key [0.0, 1.0]
  keyValue [
    0 1.0 0 0.00,
    0 1.0 0 3.14
  ]
} ...nächste Spalte

...
NavigationInfo {
  type "EXAMINE"
}

ROUTE Clock.fraction_changed
TO Interpolator.set_fraction
ROUTE Interpolator.value_changed
TO RotCube.set_rotation
```

Beispiel: Nichtlineare Geschwindigkeit

```
...
DEF Interpolator OrientationInterpolator {
  key [0.0, 0.15, 0.85, 1.0]
  keyValue [
    0 1.0 0 0.00,
    0 1.0 0 1.57,
    0 1.0 0 2.36,
    0 1.0 0 3.14
  ]
}
...

```

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
 - D.h.:
 - » 3 Werte für aktuellen Farbton
 - Beispiel:

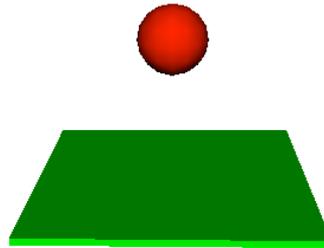
```
keyValue [
  0.5 0 0.8, #violett
  0.5 0 0.8,
  0.5 0 0.8,
  0.5 0 0.8,
  0.5 0 0.8,
  0.5 0 0.8,
  0.5 0 0.8,
  0.5 0 0.8,
  0.5 0 0.8,
  1 0 0, #rot
  1 0 0,
  0.5 0 0.8
]
```

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: Fixierung von Gegenstand

```
DEF Sensor ProximitySensor {
  size 1000 1000 1000 # Grösse des Sensorfelds
}

DEF Ball Transform {
  ...
  geometry Sphere {
    radius 0.5
  }
}

# Weitergabe der Beobachterposition an Objekt =
# Fixierung
ROUTE Sensor.position_changed TO Ball.set_translation
ROUTE Sensor.orientation_changed TO Ball.set_rotation
```

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
}
```

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
- 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, ...