

## VRML-Kurs Teil 2

### *VRML-Knoten*

Jeder VRML-Knoten kann 0 oder mehr Felder enthalten. Die Felder haben eine festgelegte Semantik, Typisierung und Default-Werte.

### *Typen in VRML 97*

Standard-Feldtypen: Bezeichnung beginnt mit SF oder MF

SF = "single field", einzelner Wert

MF = "multiple field", Array

SFNode / MFNode	VRML-Knoten
SFBool	TRUE oder FALSE
SFColor / MFColor	3 Gleitkommazahlen zwischen 0.0 und 1.0 (RGB-Farbmodell)
SFFloat / MFFloat	Gleitkommazahl(en)
SFImage	Pixel-Beschreibung einer Bitmap
SFInt32 / MFInt32	32Bit-Ganzzahlen
SFRotation / MFRotation	4 Gleitkommazahlen: 3 für die Drehachse, letzte für den Drehwinkel in Bogenmaß
SFString / MFString	Zeichenkette (utf8-Zeichensatz)
SFTime / MFTime	Anzahl Sekunden seit 1. 1. 1970, 0 Uhr, als doppelt genaue Gleitkommazahl
SFVec2f / MFVec2f	2 Gleitkommazahlen als 2D-Vektor (bzw. Array solcher Vektoren)
SFVec3f / MFVec3f	3 Gleitkommazahlen als 3D-Vektor (bzw. Array solcher Vektoren)

MF-Werte werden in eckige Klammern [ ] eingeschlossen und innerhalb der eckigen Klammern durch Kommata oder Leerzeichen voneinander getrennt. Bei genau einem Wert können die eckigen Klammern auch weggelassen werden.

Beispiel:

Koordinatenknoten haben ein Feld vom Typ MFVec3f. Der Inhalt ist eine Liste von 3D-Ortsvektoren.

**Coordinate**

```
{
  point [ x1 y1 z1, ..., xn yn zn ]
}
```

*Elementarknoten in VRML 97:*

<i>Knoten</i>	<i>Felder</i>	<i>Feldtyp</i>	<i>Bedeutung</i>
<b>Shape</b>	<b>appearance</b> <b>geometry</b>	SFNode SFNode	Materialdaten Geometrie-Knoten (Primitiv-Obj., Mengen...)
<b>Coordinate</b>	<b>point</b>	MFVec3f	Liste v. 3D-Koord.
<b>Normal</b>	<b>point</b>	MFVec3f	Liste v. 3D-Koord. (zur Festlegung v. Normalenvektoren)

Zentral sind *Gruppen-* und *Transformations-Knoten* (für die Konstruktion des Szenengraphen). Sie gruppieren bzw. transformieren beliebige Unterbäume im Szenengraphen:

<b>Group</b>	<b>children</b>	MFNode	Array der Kind-Knoten
	<b>addChildren</b>	MFNode	für eventIn
	<b>removeChildren</b>	MFNode	für eventIn
<b>Transform</b>	<b>center</b>	SFVec3f	Zentr. für Rot. u. Skalierung
	<b>scale</b>	SFVec3f	Sk.-faktoren
	<b>scaleOrientation</b>	SFRotation	Rot. für scale
	<b>rotation</b>	SFRotation	Rotation
	<b>translation</b>	SFVec3f	Translation
	<b>children</b>	MFNode	Kind-Knoten
	<b>addChildren</b>	MFNode	für eventIn
	<b>removeChildren</b>	MFNode	für eventIn
	<b>bboxCenter</b>	SFVec3f	Zentr. d. bbox
<b>bboxSize</b>	SFVec3f	bbox-Größe	

In VRML 1.0 gibt es auch die Möglichkeit, Transformationen direkt über eine Matrix zu spezifizieren:

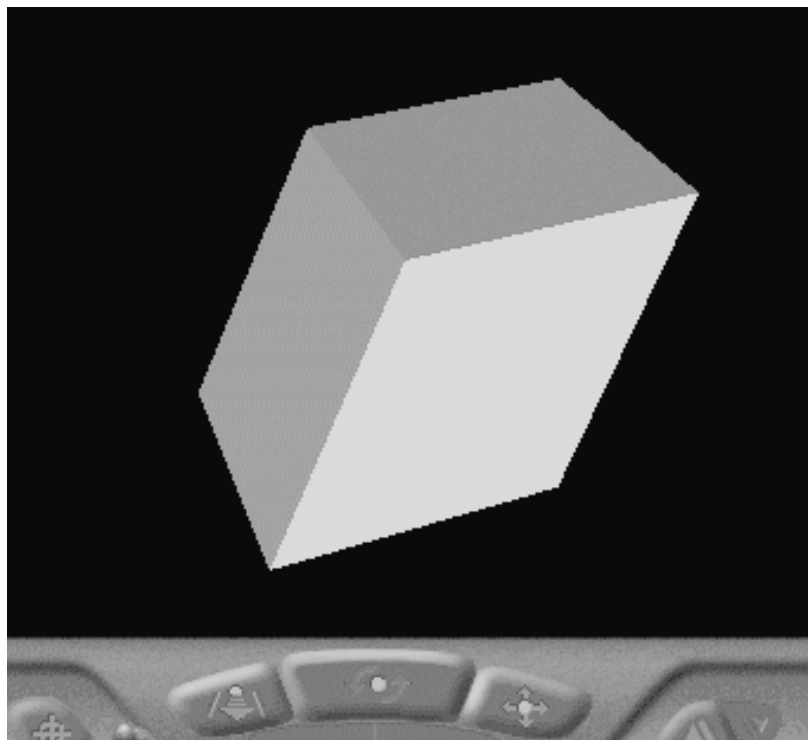
<b>MatrixTransform</b>	<b>matrix</b>	MFFloat (16 Einträge)	Transf.-matrix (transponiert)
------------------------	---------------	--------------------------	----------------------------------

Beispiel:

Anwendung einer Scherung auf einen Würfel

```
#VRML V1.0 ascii
MatrixTransform
{
  matrix 1 0 0 0
         0.5 1 0 0
         0 0 1 0
         0 0 0 1
}
Cube { }
```

Ergebnis:



*Grafische Primitive in VRML 97:*

<b>Box</b>	<b>size</b>	SFVec3f	Kantenlängen eines geschlossenen Quaders
<b>Cone</b>	<b>bottomRadius</b> <b>height</b> <b>side</b> <b>bottom</b>	SFFloat SFFloat SFBool SFBool	Radius des Bodens Höhe des Kegels Sichtbarkeit: Mantel Sichtbarkeit: Boden
<b>Cylinder</b>	<b>bottomRadius</b> <b>height</b> <b>side</b> <b>top</b> <b>bottom</b>	SFFloat SFFloat SFBool SFBool SFBool	Radius Höhe des Zylinders Sichtb.: Mantel Sichtb.: Deckel Sichtb.: Boden
<b>Sphere</b>	<b>radius</b>	SFFloat	Radius der Kugel
<b>Text</b>	<b>string</b>  <b>fontStyle</b> <b>length</b> <b>maxExtent</b>	MFString  SFNode MFFloat SFFloat	1 oder mehrere Zeichenketten Font-Spezifikation max. Länge max. phys. Länge

*Knoten für Punktmengen und boundary repr. in VRML 97:*

Punktmenge (ohne Flächen und Kanten):

<b>PointSet</b>	<b>coord</b> <b>color</b>	SFNode SFNode	Koordinatenknoten Farbknoten
-----------------	------------------------------	------------------	---------------------------------

darin der Farbknoten:

<b>Color</b>	<b>color</b>	MFCColor	RGB-Spezifikation(en)
--------------	--------------	----------	-----------------------

Indizierte Kantenmenge:

<b>IndexedLineSet</b>	<b>coord</b> <b>coordIndex</b> <b>set_coordIndex</b> <b>color</b> <b>colorIndex</b> <b>set_colorIndex</b> <b>colorPerVertex</b>	SFNode MFInt32 MFInt32 SFNode MFInt32 MFInt32 SFBool	Koord.knoten Polylinien für eventIn Farbzuordn. Farbzuordn. für eventIn Farbe bez. auf Ecken (sonst aufKanten)
-----------------------	---------------------------------------------------------------------------------------------------------------------------------------------------	------------------------------------------------------------------------	----------------------------------------------------------------------------------------------------------------------------------------

## Indizierte Flächenmenge:

<b>IndexedFaceSet</b>	<b>coord</b>	SFNode	Koord.knoten
	<b>coordIndex</b>	MFInt32	Polygone
	<b>set_coordIndex</b>	MFInt32	für eventl.
	<b>color</b>	SFNode	Farbzuordn.
	<b>colorIndex</b>	MFInt32	Farbzuordn.
	<b>set_colorIndex</b>	MFInt32	für eventl.
	<b>colorPerVertex</b>	SFBool	Farbe bez. auf Ecken (sonst auf Flächen)

Die Indextabelle (**coordIndex**) enthält die Indices (Zählung beginnt bei 0) der zu einer Polylinie bzw. zu einem geschlossenen Polygon gehörenden Ecken. Trennung mehrerer Polylinien / Polygone durch den Eintrag "-1":

**coordIndex [1,5,4,2, -1, 0,3,7,6, -1, 1,5,7,6]**  
= 3 Vierecke

Orientierung ist wichtig – für die Normalenvektoren gilt die "Rechte-Hand-Regel"; die Rückseite eines Polygons (entgegengesetzt zum Normalenvektor) ist unsichtbar.

## Beispiele:

### Konstruktion einer Pyramide als Punktmenge

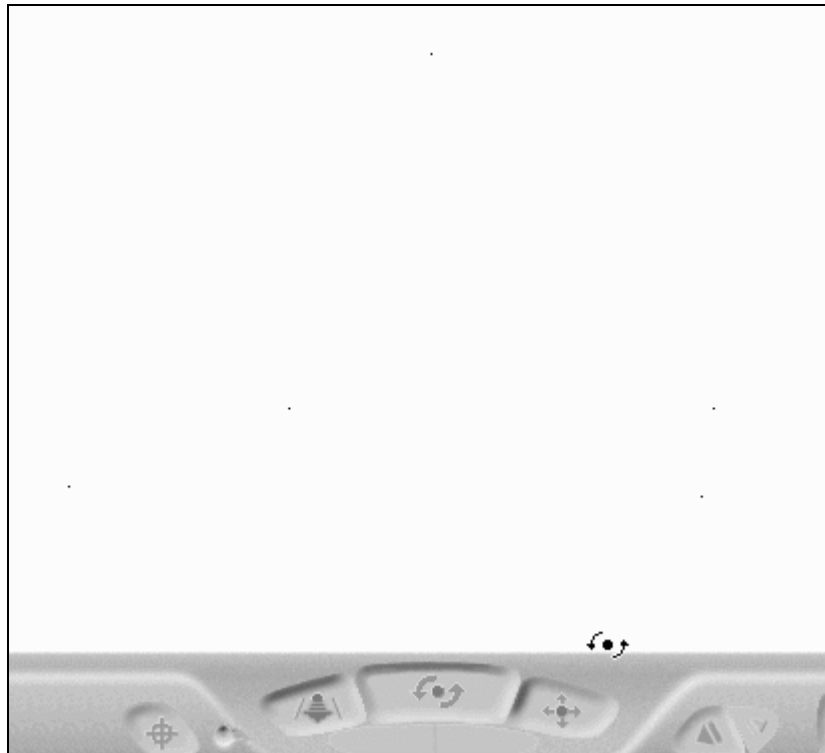
```
#VRML V2.0 utf8
# pyrpkt.wrl
Shape
{
  geometry PointSet
  {
    coord Coordinate
    {
      point [ -2, 0, -2
              2, 0, -2
              -2, 0, 2
              2, 0, 2
              0, 3, 0 # Spitze
```

```

    ]
  }
}

```

Ergebnis (Bild invertiert, da Eckpunkte sonst kaum sichtbar):



Pyramide als durch Eckenindices definierte Kantenmenge:

```

#VRML V2.0 utf8
# pyrlin3.wrl
Shape
{
  geometry IndexedLineSet
  {
    coord Coordinate
    {
      point [ -2 0 -2 # Ecke 0
              2 0 -2 # Ecke 1
              -2 0 2 # Ecke 2
              2 0 2 # Ecke 3

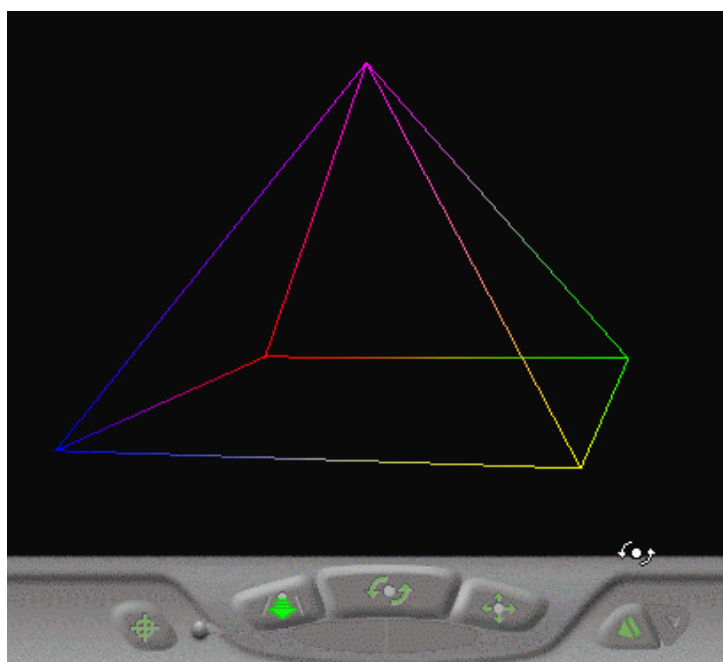
```

```

                                0 3 0 # Ecke 4 = Spitze
                                ]
                                }
coordIndex [ 0 1 -1,
              1 3 -1,
              3 2 -1,
              2 0 -1,
              0 4 -1,
              1 4 -1,
              3 4 -1,
              2 4
              ]
color Color
{
  color [ 1 0 0,
          0 1 0,
          0 0 1,
          1 1 0,
          1 0 1 ]
}
}
}

```

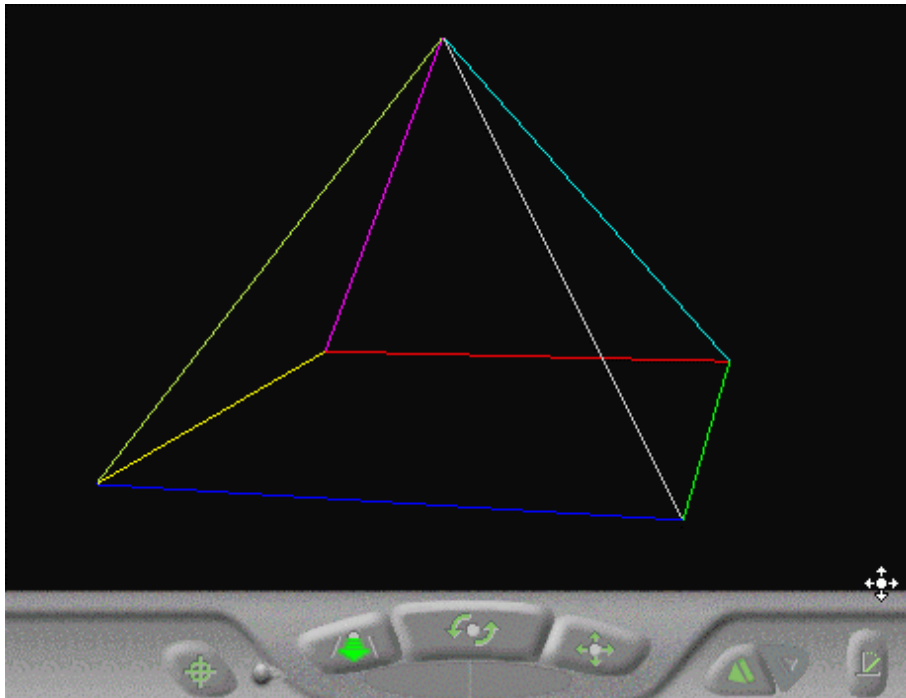
Ergebnis:



Die Farben werden hier zwischen den Ecken interpoliert.  
Direkte Farbzuzuordnung zu den Kanten durch Einfügen von

`colorPerVertex FALSE`

in den `IndexedLineSet`-Knoten:



auch möglich: Polylinien (mehr als 2 Punkte) anstatt Kanten

```
coordIndex [ 4 1 0 -1,  
            4 3 1 -1,  
            4 2 3 -1,  
            4 0 2 -1,  
            2 0 1 3    # Boden  
            ]
```



Pyramide als Körper mit durch Punktindices definierten Facetten:

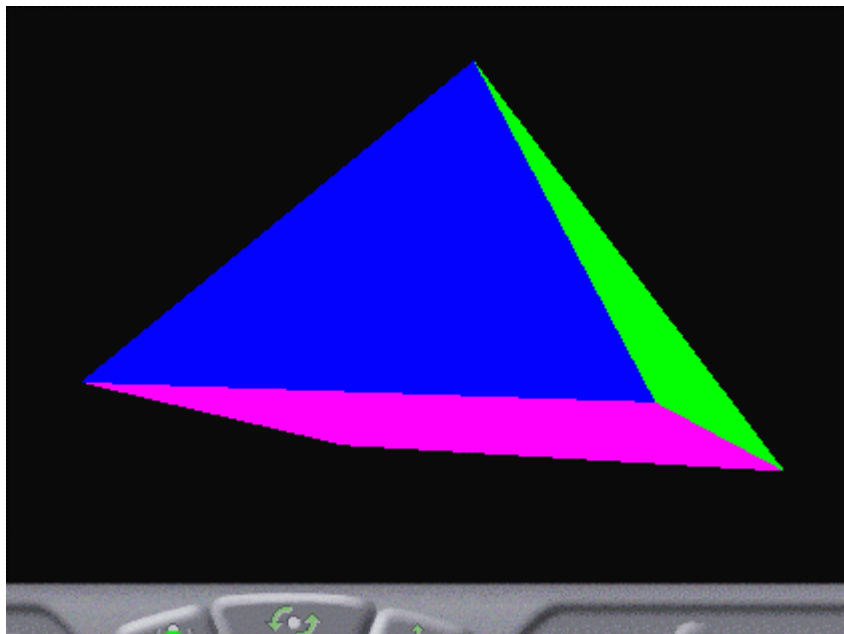
```
#VRML V2.0 utf8
# pyrface1.wrl
Shape
{
  geometry IndexedFaceSet
  {
    coord Coordinate
    {
      point [ -2 0 -2 # Ecke 0
              2 0 -2 # Ecke 1
              -2 0 2 # Ecke 2
              2 0 2 # Ecke 3
              0 3 0 # Ecke 4 = Spitze
            ]
    }
    coordIndex [ 4 1 0 -1,
                 4 3 1 -1,
                 4 2 3 -1,
                 4 0 2 -1,
                 2 0 1 3 # Boden
               ]
    color Color
    {
      color [ 1 0 0,
              0 1 0,
              0 0 1,
              1 1 0,
              1 0 1 ]
    }
  }
}
```

Ergebnis:



auch hier Farbinterpolation zwischen den Eckpunkten – direkte  
Farbzuordnung zu den Facetten wird durch den Switch  
**colorPerVertex FALSE**

eingestellt:



*Extrusion*: Spur eines in der xz-Ebene def. Polygons entlang einer Raumkurve

<b>Extrusion</b>	<b>crossSection</b>	MFVec2f	Polygon
	<b>spine</b>	MFVec3f	Stützpunkte der Kurve
	<b>endCap</b>	SFBool	Deckel exist.
	<b>solid</b>	SFBool	Fläche beidseitig sichtbar
	<b>scale</b>	MFVec2f	Skalierungs-Array
	<b>orientation</b>	MFRotation	Drehungen entl. Kurve

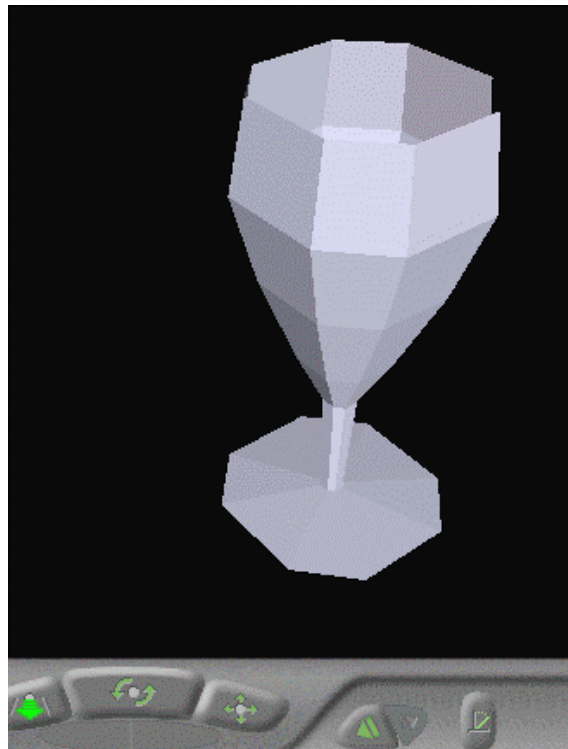
Beispiel:

```
#VRML V2.0 utf8
# wein1.wrl
```

```
Shape
{
  geometry Extrusion
  {
    endCap FALSE
    solid FALSE
    crossSection [ -1 -3, -3 -1, -3 1, -1 3,
                  1 3, 3 1, 3 -1, 1 -3, -1 -3 ]
    spine [ 0 0 0, 0 0.6 0, 0 7 0, 0 10 0,
            0 15 0, 0 20 0, 0 25 0 ]
    scale [ 2 2, 0.2 0.2, 0.3 0.3, 0.8 0.8,
            1.5 1.5, 2 2, 1.8 1.8 ]
  }
  appearance Appearance
  {
    material Material
    { diffuseColor 0.9 0.9 1 }
  }
}
```

achteckige Grundfläche, Extrusionskurve (*spine*) mit 7 Stützpunkten

Ergebnis:



Modifikation: Drehung während der Extrusion  
– füge in den Extrusion-Knoten für jeden Stützpunkt eine Orientierungsspezifikation ein:

```
orientation [ 0 1 0 0, 0 1 0 0, 0 1 0 0.3,  
             0 1 0 0.6, 0 1 0 0.9, 0 1 0 1.2,  
             0 1 0 1.5 ]
```

Ergebnis:



## Höhengitter: Erhebungsgitter über der xz-Ebene

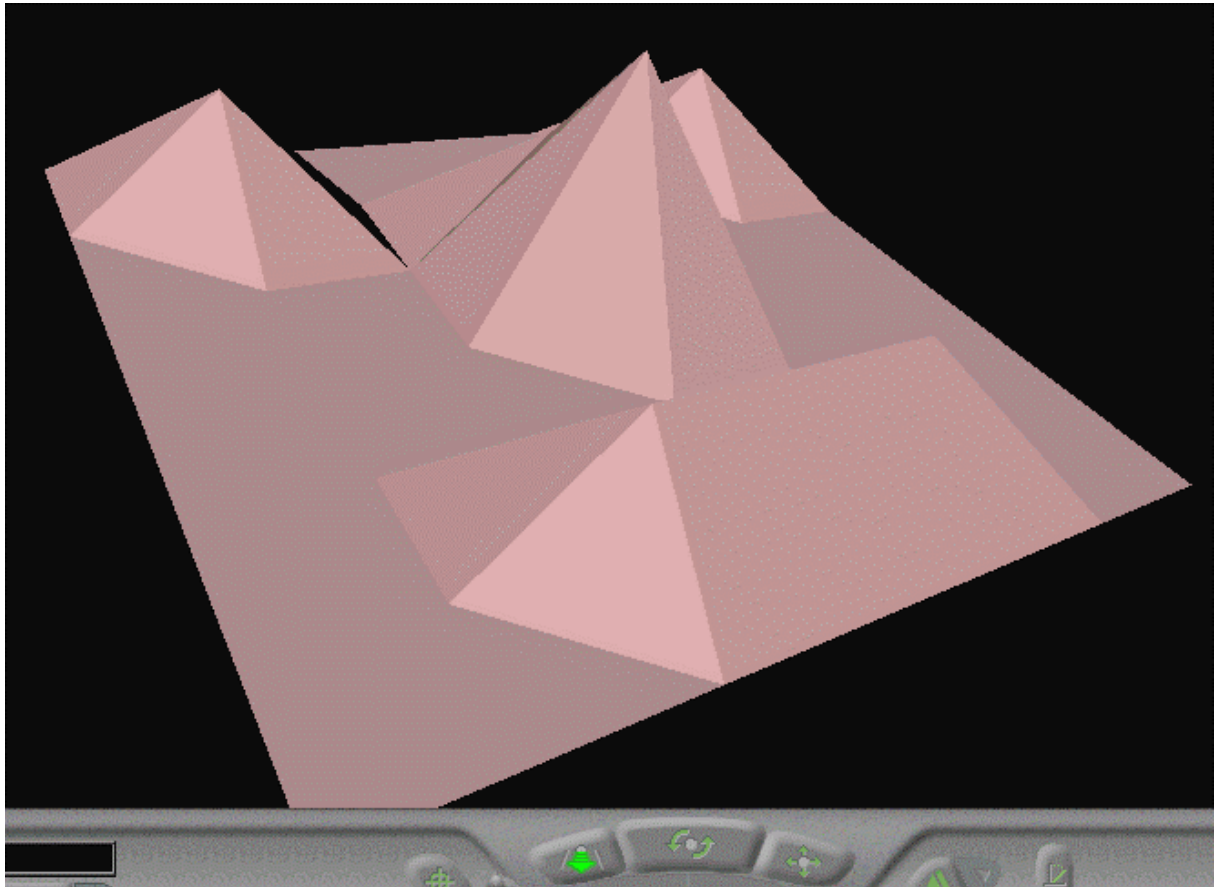
<b>ElevationGrid</b>	<b>xDimension</b>	SFInt32	Anz. Pkte in x-Richtung
	<b>xSpacing</b>	SFFloat	Abstand
	<b>zDimension</b>	SFInt32	Anz. Pkte in y-Richtung
	<b>zSpacing</b>	SFFloat	Abstand
	<b>height</b>	MFFloat	Höhenwerte-Array
	<b>colorPerVertex</b>	SFBool	Farb-Flag
	<b>creaseAngle</b>	SFFloat	Kantenschärfe
	<b>ccw</b>	SFBool	counter-clockwise

Beispiel:

```
#VRML V2.0 utf8
# landsc1.wrl
```

```
Shape
{
  geometry ElevationGrid
  {
    xDimension 7
    zDimension 7
    xSpacing 1
    zSpacing 1
    height [ 0 0 0 0 0 0 0
            0 1 0 0 0 1 0
            0 0 0 1 0 0 0
            0 0 0 2 0 0 0
            0 0 0 0 0 0 0
            0 0 1 1 1 0 0
            0 0 0 0 0 0 0 ]
  }
  appearance Appearance
  {
    material Material
    { diffuseColor 0.9 0.7 0.7 }
  }
}
```

Ergebnis:



*Hintergrund-Knoten:*

<b>Background</b>	<b>groundColor</b>	MFCColor	Farbwerte
	<b>groundAngle</b>	MFFloat	Winkel für Bodenfarben
	<b>skyColor</b>	MFCColor	Farbwerte
	<b>skyAngle</b>	MFFloat	Winkel für Himmelfarben
	<b>backUrl</b>	SFString	URL für hinteres,
	<b>bottomUrl</b>	SFString	unteres usw.
	<b>frontUrl</b>	SFString	Hintergrundbild
	<b>leftUrl</b>	SFString	
	<b>rightUrl</b>	SFString	
	<b>topUrl</b>	SFString	
	<b>set_bind</b>	SFBool	für eventIn
	<b>isBound</b>	SFBool	für eventOut

Anwendung auf das letzte Beispiel

wir hängen ans Ende an:

Background

```
{
  skyAngle [ 1.6 ]
  skyColor [ 0 0 1, 0.4 0.4 1 ]
  groundAngle [ 1.6 ]
  groundColor [ 0.8 0.8 0.8, 0.01 0.025 0.001 ]
  frontUrl "wiessee.jpg"
  backUrl "wiessee.jpg"
  leftUrl "wiessee.jpg"
  rightUrl "wiessee.jpg"
}
```

Ergebnis:



### Hyperlink zu anderen VRML-Dateien:

<b>Inline</b>	<b>description</b>	SFString	Targetbeschreibung
	<b>parameter</b>	MFString	Parameter
	<b>url</b>	MFString	Liste der Targets

### Hyperlink zu anderen Dateien (nicht notw. VRML):

<b>Anchor</b>	<b>description</b>	SFString	Targetbeschreibung
	<b>parameter</b>	MFString	Parameter
	<b>url</b>	MFString	Liste der Targets
	<b>children</b>	MFNode	Feld für Kindknoten
	<b>addChildren</b>	MFNode	für eventIn
	<b>removeChildren</b>	MFNode	für eventIn
	<b>bboxCenter</b>	SFVec3f	Zentr. der bounding box
	<b>bboxSize</b>	SFVec3f	Größe der b. box

Beispiel:

2 Dateien, die wechselseitig verlinkt sind.

Erste Datei:

```
#VRML V2.0 utf8
# anchor1.wrl
```

```
Anchor
{
  url "anchor2.wrl"
  description "Eintritt in eine neue Welt"
  children
  [
    Shape
    {
      geometry Box { size 1 1 2 }
      appearance Appearance
      {
        texture ImageTexture
        { url "wiessee.jpg" }
      }
    }
  ]
}
```

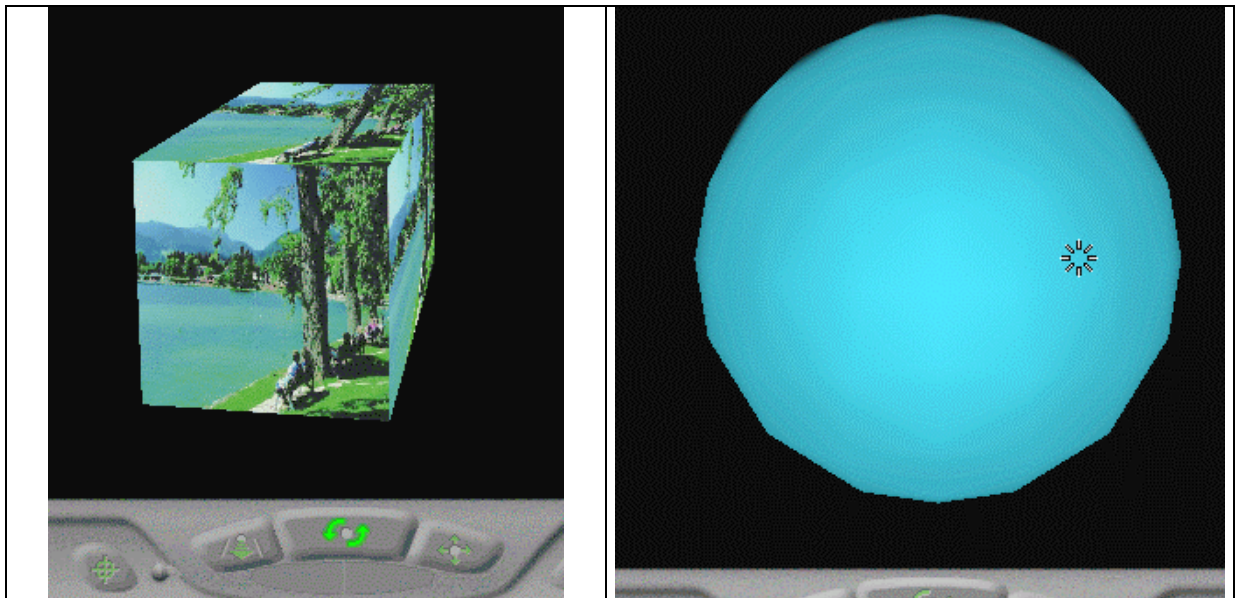


Zweite Datei:

```
#VRML V2.0 utf8
# anchor2.wrl
```

```
Anchor
{
  url "anchor1.wrl"
  description "Zurück in die alte Welt"
  children
  [
    Shape
    {
      geometry Sphere { radius 3 }
      appearance Appearance
      {
        material Material
        { diffuseColor 0.3 0.9 1 }
      }
    }
  ]
}
```

Ergebnis:



Die "Teleportation" von einer Welt in die andere erfolgt durch Anklicken des Anker-Objekts mit der Maus. (Der Mauscursor verändert seine Gestalt über dem Anker-Objekt, siehe rechts.)