

**1. Einfaches VRML-Programm**

#VRML V2.0 utf8

```
Shape {
  appearance Appearance {
    material Material {}
  }
  geometry Box {}
}
```

#Gestaltknoten  
#Subknoten (Beschreibung der Oberfläche)  
#Subknoten (Beschreibung des Materials)  
#Geometrie-knoten (hier: Quader)

**2. Vier vordefinierte geometrische Grundformen**

- a. Quader: `geometry Box {  
size 1.0 1.0 1.0  
}` Reihenfolge x, y, z
- b. Kegel: `geometry Cone {  
bottomRadius 1.0 height 1.0  
}`
- c. Zylinder: `geometry Cylinder {  
radius 1.0 height 1.0  
}`
- d. Kugel: `geometry Sphere {  
radius 1.0  
}`

**3. VRML-Programm mit Farben**

#VRML V2.0 utf8

```
Background {skyColor 0.0 0.0 1.0} #Hintergrundfarbe (hier: blau)

Shape {
  appearance Appearance {
    material Material {diffuseColor 1.0 1.0 0.0} #Subknoten (Beschreibung des
  } #Materials, Farbe gelb)
  geometry Box {}
}
```

**4. VRML Programm mit verschobenem Körper**

#VRML V2.0 utf8

```
Background {skyColor 1.0 1.0 1.0} #Hintergrundfarbe (hier: weiß)

Transform{ #Gruppenknoten (hier: "Transform" zum Verschieben)
  children [ #untergeordneter Knoten (hier: „Children“, Liste von
    Shape { #Kindknoten
      appearance Appearance {
        material Material {diffuseColor 1.0 1.0 0.0}
      }
      geometry Box {}
    }
  ]
  translation 2.0 0.0 0.0 #x, y, z Verschiebung
}
```

## 5. Mehrfachverwendung von Knoten

#VRML V2.0 utf8

#3-dimensionales Pluszeichen

#Definition eines Zylinders in Größe und Farbe, ergibt auch ersten Teil des Pluszeichens

```
DEF Zylinder Shape {                                     #Benennung Knoten (hier: "Zylinder" nach DEF)
    appearance Appearance {
        material Material {
            diffuseColor 0.8 0.8 0.8
        }
    }
    geometry Cylinder {
        radius 0.5
        height 5.0
    }
}
```

#Wiederaufruf der Definition, zweiter Teil des Pluszeichens

```
Transform {
    children USE Zylinder                               #Wiederaufruf Knoten "Zylinder" mit USE
    rotation 0.0 0.1 1.0 1.0472                       #Reihenfolge: x, y, z, Winkel
}
```

#Wiederaufruf der Definition, dritter Teil des Pluszeichens

```
Transform {
    children USE Zylinder
    rotation 0.0 0.0 1.0 2.0944                       #Reihenfolge: x, y, z, Winkel
}
```

### Literatur:

- Das Einsteigerseminar VRML, Rolf Däßler, bhv-Verlag 1999 (Buch)  
oder:  
<http://fabdq.fh-potsdam.de/bhv/> (Internet)
- 3D mit VRML von Uwe Debacher  
<http://www.debacher.de/vrml/>
- und viele andere Quellen

**6. Programm mit Animation**

```

#VRML V2.0 utf8

#Hintergrund
Background {skyColor 1.0 1.0 1.0}

#Zeittakt
DEF Uhr TimeSensor {
    cycleInterval 5.0
    loop TRUE
}
#Steuert Animation
#Länge des Zeitintervalls
#Zeitschleife ist eingeschaltet

#Bausteine
DEF Zylinder Transform{
    children [
        Shape {
            appearance Appearance {
                material Material {
                    diffuseColor 0.0 0.0 1.0
                }
            }
            geometry Cylinder {height 3.0}
        }
    ]
}

DEF Quader Transform{
    children [
        Shape {
            appearance Appearance {
                material Material {
                    diffuseColor 1.0 1.0 0.0
                }
            }
            geometry Box {size 5.0 2.0 2.0}
        }
    ]
    translation 0.0 2.5 0.0
}

#Berechnung Animation (Interpolator)
DEF PI_Quader PositionInterpolator {
    keyValue [
        0.0 2.5 0.0,
        0.0 2.5 0.0,
        0.0 2.5 0.0,
        -5.0 2.5 0.0,
        -5.0 0.0 0.0
    ]
    key [0.0, 0.25, 0.5, 0.75, 1.0]
}
#Berechnung der Animation aus Schlüsselwerten,
#wird durch Zeitgeber synchronisiert
#Schlüsselpositionen in x-, y-, z-Ebene, werden
#nacheinander eingenommen

#relativer Zeitpunkt für jeweilige
#Schlüsselposition

#Routen
ROUTE Uhr.fraction_changed TO PI_Quader.set_fraction
ROUTE PI_Quader.value_changed TO Quader.set_translation
#Übergabe relativer Zeit an Interpolator
#Übergabe interpolierter Positionswerte
#an Transform-Knoten

```