



# VRML Programmierung 2

Weitere Programmiermöglichkeiten in VRML, Vorgehensweise und Methodik



# Shapes...

- **Box**

- size `<x-size> <y-size> <z-size>`

- **Sphere**

- radius `<value>`

- **Cone**

- bottomRadius `<value>`
- Height `<value>`
- bottom `<FALSE|TRUE>`

- **Cylinder:** `radius, height, top`



# Mehr zu material{ }

- `diffuseColor`  
Die normale Farbe des Objektes
- `specularColor`  
Die Farbe der Lichtreflexe
- `emissiveColor`  
Die Farbe, in der das Objekt selbst leuchtet
- `ambientIntensity`  
Die Menge reflektierten Umgebungs-Lichtes
- `shininess`  
Wie stark das Objekt reflektiert
- `transparency`  
Transparenz des Objektes



# Beispiel Farben

```
Shape {  
    appearance Appearance {  
        material Material {  
            emissiveColor 0 0.8 0  
            transparency 0.5  
        }  
    }  
    geometry Box {  
    }  
}
```



# Spezielle Farbwerte

● Weiss:	1	1	1
● Schwarz:	0	0	0
● Rot:	1	0	0
● Grün:	0	1	0
● Blau:	0	0	1
● Grau:	0.5	0.5	0.5

# Texturen (statt material{ })

- Muster (Texturen) ermöglichen, realistische Oberflächen darzustellen
- Beispiel:

```
Appearance {  
    texture ImageTexture {  
        url "brick.jpg"  
        repeatT TRUE      # vertikal  
        repeatsS TRUE     # horizontal  
    }  
}
```

# MPEGs als Texturen

- MovieTexture bringt ein MPEG1 File auf die Oberfläche eines Objektes, zus. Felder:
  - speed  
Geschwindigkeits-Faktor (1=Normal, 2=Doppelt usw.).
  - loop  
TRUE wenn Wiederholung, FALSE wenn nicht.
  - startTime  
Start-Zeitpunkt (Sekunden seit 1.1.1970 ;-)
  - stopTime  
Stop-Zeitpunkt (dto.)
- **Wer es zum Laufen kriegt, soll sich melden...**



# Text ist ebenfalls möglich...

```
geometry Text {  
    string ["Hello", "World"]  
    fontStyle FontStyle {  
        size 0.8  
        justify "MIDDLE"  
    }  
    maxExtent 5  
    length [3, 3]  
}
```



# Sprachelemente für Abstraktionen

- Grössere Projekte müssen wie üblich bearbeitet werden:
  - Anforderungsanalyse / Spezifikation,
  - Strukturierung in Abstraktions-Elemente,
  - Implementierung der Elemente,
  - Implementierung der Struktur.
- Einfache Variante: DEF / USE
- Höhere Form: PROTO u. EXTERNPROTO

# Wiederverwendung PROTO

```
PROTO VBox [  
    field SFCOLOR boxColour 1 0 0  
]  
  
{  
    Shape {  
        appearance Appearance {  
            material Material {  
                diffuseColor IS boxColour  
            }  
        }  
        geometry Box {  
        }  
    }  
}
```

```
--> VBox { boxColour 0 1 0 }
```

# Wiederverwendung EXTERN...

- PROTO Definition ist in einem anderen File

```
EXTERNPROTO VBox [  
    field SFCOLOR boxColour  
]  
  
[  
    "proto.wrl"          # Alternativen 1 + 2  
    "http://www.mydomain.com/protos/proto.wrl"  
]
```

- Wenn mehrere Prototypen in einem .wrl File enthalten sind, benutzt man Targets:

```
"proto.wrl#VBox"
```



# Vor-Übungen zur Belegarbeit

- Setzen Sie Ihre einfachen Welten fort, die aus den Grundgeometrien “Sphere”, “Cone”, “Box” und “Cylinder” bestehen.
- Nehmen Sie Elemente zur Wiederverwendung hinzu!
- Experimentieren Sie mit **Text** und **\*Texture Nodes**!
- Informieren Sie sich über Events & Routen!



# Das nächste Projekt (Beleg)

- Ziel: Komplexe(re) virtuelle Welt
- Vorgehensweise:
  - Thema wählen,
  - Req.-Definition / Spezifikation,
  - Abstrahieren / Strukturieren der Bausteine,
  - Implementierung.
- Dokumentation der Implementierung mit Begründung der Design-Entscheidungen.



# Nochmal das Tutorial:

[http://web3d.vapourtech.com/  
tutorials/vrml97/index.html](http://web3d.vapourtech.com/tutorials/vrml97/index.html)