VRML 2.0 und Behaviour

Seminar im Hauptstudium
Computergraphik, Visualisierung und geometrische Modellierung

Matthias Ott

10. Dezember 1996

Inhaltsverzeichnis

1 Definition

Der Grundgedanke zur Entwicklung von VRML war die Visualisierung von virtuellen Welten im Internet und damit einhergehend die Etablierung eines Graphikstandards. Dabei sollte die Beschreibungssprache folgende drei Grundvoraussetzungen erfüllen:

Sie sollte unabhängig von der verwendeten Hardwareplattform sein, sich beliebig erweitern lassen und auch mit langsamen Verbindungen (14.400 Bit/s - Modem) noch hinreichend benutzbar sein.

2 Ziele des Sprachstandards

nach der VRML Architecture Group (VAG):

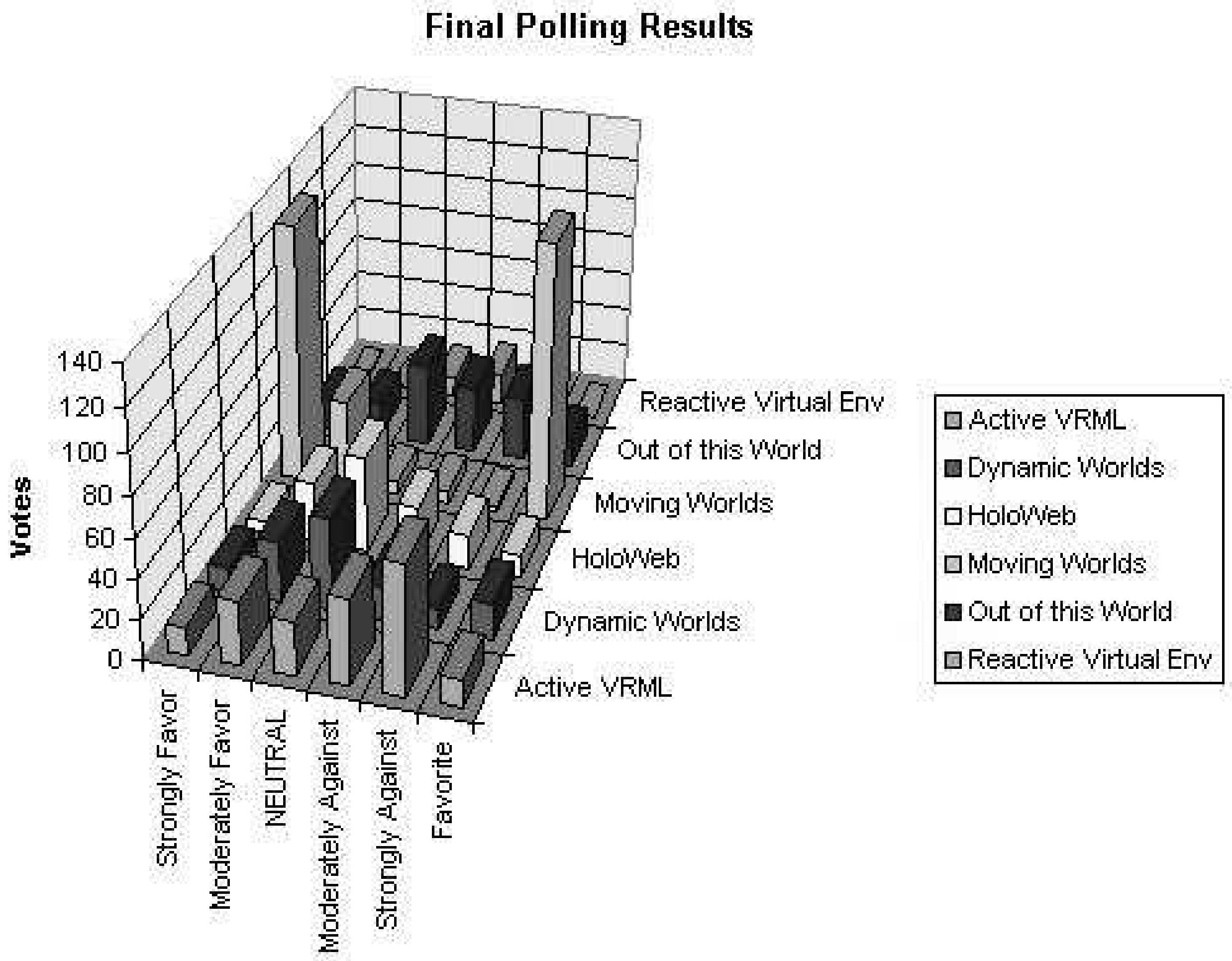
- Authorability
- Completeness
- Composability
- Extensibility
- Implementability
- Multi-user potential
- Orthogonality
- Performance
- Scalability
- Standard practice
- Well-structured

3 Moving Worlds und Active VRML

Moving Worlds erweitert VRML 1.0 um folgende Interaktivität:

- Erweiterte statische Weltbeschreibung
- Interaktion
- Animation
- Scripting
- Prototyping

Active VRML wurde von Microsoft als Erweiterung von VRML 1.0 vorgeschlagen, unterlag aber in der abschließenden Abstimmung Moving Worlds.



4 Beschreiben einer VRML Szene

4.1 Einheiten

- Entfernungen sind in Meter
- Winkel sind in Radian
- die Zeit steigt monoton
- keine Unterscheidung zwischen diskreten und kontinuierlichen Veränderungen

4.2 Erstellen eines Szenengraph

- Die Szene wird durch ein ASCII-File erzeugt:
  `#VRML V2.0 utf8`
- # leitet Kommentare ein

Gruppierungsknoten:

- Transform
  `rotation, scale, scaleOrientation, translation`
- Group
- Inline
- Anchor

```
Anchor {
    url "http://foo.com/someScene.wrl\#Overview"
    children Shape { geometry Box {} }
}
```

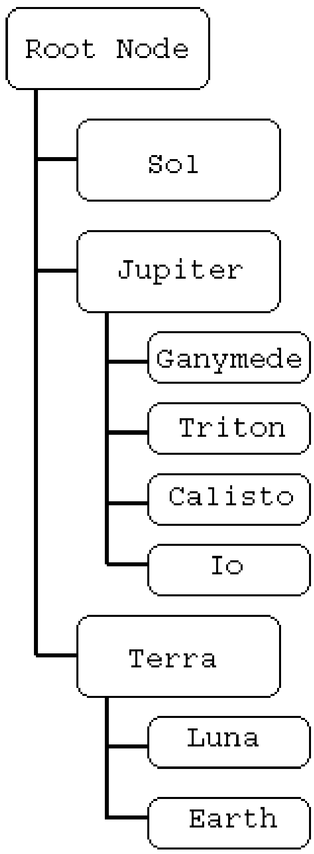
- Billboard

Active VRML:

- *Gliederung durch Vereinigung*
  `solarSystem = sun union earth`
- *lokaler Namensraum durch* `let` *möglich*

```
let
    earth = import("sphere.wrl");
    sun = import("sphere.wrl");
in
    sun union earth
```

- *Translationen durch explizite Funktionen*
  - `scale3()`
  - `translate()`
  - `moveXby2()`



4.3 Geometrie

Grundprimitive:

- Box
- Cone
- Cylinder
- Sphere
- Text
- ElevationGrid
- Extrusion
- IndexedFaceSet
- IndexedLineSet
- PointSet

4.4 Erscheinungsbild

- Material
  { `ambientIntensity`, `diffuseColor`, `emissiveColor`, `shininess`, `specularColor`, `transparency` }
- ImageTexture
- PixelTexture
- MovieTexture
- Appearance
- Shape

- *Active VRML: durch Funktionen zugewiesen*

```
sphere = import("sphere.wrl");
earthMap = import("earth-map.gif");
earthTexture = texture(earthMap);
unitEarth = earthTexture(sphere);
halfScale = scale3(0.5);
earth = apply(halfScale, unitEarth);
sun = diffuseColor(yellow)(sphere);
```

4.5 Instanziierung

Naming Konzept

- DEF
- USE
- *Active VRML: explizites Naming mit NAME = OBJEKT*

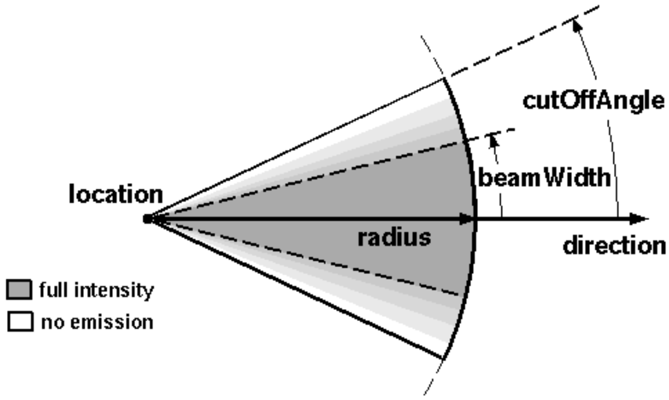
5 Weitere Eigenschaften

5.1 Lichtquellen

- DirectionalLight
- PointLight
- SpotLight

Diese beinhalten u.a. die Felder:

- intensity
- color
- ambientIntensity




```
PointLight {
    location 0 0 0
    radius 100
    intensity 1
    color 1 1 1
    ambientIntensity 0
    attenuation 1 0 0
    on TRUE
    cutOffAngle 0.785398
    beamWidth 1.570796
    direction 0 0 -1
}
```

5.2 Text

- Texte sind zweiseitig und flach
- das Aussehen basiert auf dem `FontStyle` Knoten

5.3 Weitere Knoten

- Boden- und Himmelhintergrund mit `Background`
- entfernte Berge und Wolken mit `Background`
- entfernte Objekte in Nebel

```
Fog {
    color 1 1 1
    fogType "LINEAR"
    visibilityRange 10.0
}
```

- Sound

```
Sound {
    direction 0 0 1
    location 0 0 0
    source AUDIOCLIP
    spazialize TRUE
}
```

- Viewpoint
- NavigationInfo

6 Zeitliche Veränderungen

6.1 Animation

Animationen werden durch Keyframes definiert, die durch Interpolatoren ausgewertet werden.

- ColorInterpolator
- CoordinateInterpolator
- NormalInterpolator
- OrientationInterpolator
- PositionInterpolator
- ScalarInterpolator

```
ScalarInterpolator {
    key [ 0.0, 5.0, 10.0 ]
    value [ 11.0, 99.0, 33.0 ]
}
```

- *Active VRML: Animationen werden durch Formeln erzeugt.*
  - *derivative()*
  - *integral()*
  - *sin()*
  - *cos()*

6.2 Sensoren

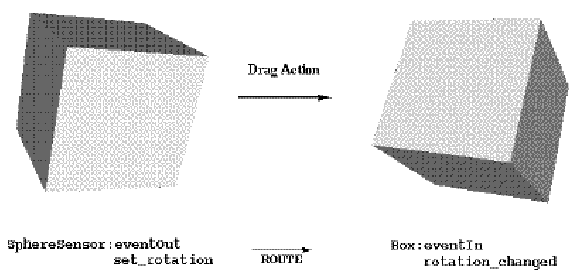
Hiermit ist eine direkte Interaktion mit Objekten ist möglich. Die Sensorknoten erzeugen Ereignisse, die an die Objekte weitergeleitet werden.

- Geometrische Sensoren
  - ProximitySensor
  - VisibilitySensor
  - TouchSensor
  - CylinderSensor
  - PlaneSensor
  - SphereSensor
  - Collision
- Zeitliche Sensoren
  - TimeSensor
- *Active VRML: es gibt eine große Anzahl von Ereignissen:* `leftButtonPress`, `leftButtonRelease`, `keyPress`, usw.

6.3 Ereignisse

- beginnen oder enden meist mit `set_` oder `_changed`
  `ROUTE Name.eventOutName_changed TO Name.set_eventInName`
- werden durch Routen zwischen verschiedenen Knoten weitergeleitet

```
DEF CLICKER Touchsensor { enabled TRUE }
DEF LIGHT DirectionalLight { on FALSE }
ROUTE CLICKER.enabled_changed TO LIGHT.set_on
```



Active VRML:

- *Ereignisse werden durch* `=>` *weitergeleitet:*

```
twoPhase = red until LBP => green
```

- *Bedingungen werden mit* `predicate` *abgefragt*

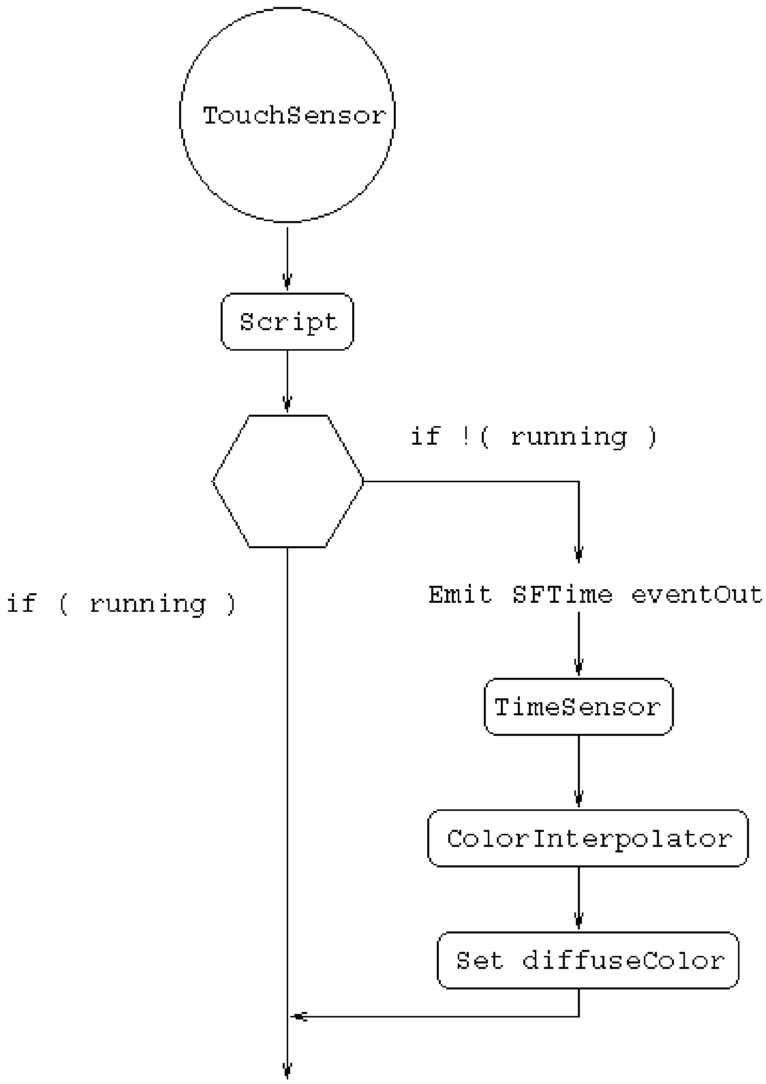
```
score(current) =
    current until
        scored => score(current+1)
        | predicate(current = maxScore) => current
```

6.4 Scripting

- Scripting ist für fast für jede Bewegung zuständig
- das Script erhält Werte von Sensoren und generiert daraus Ereignisse, die andere Knoten verändern
- als Scriptsprache ist v.a. Java und Javascript vorgesehen
- spezielle Methoden:
  - `initialize`
  - `shutdown`
  - `processEvent`

```
Script {
    eventIn SFBool start
    url [ "http://foo.com/fooBar.class"]
}
```

- *Active VRML: es gibt eine integriert Scriptsprache*



Literatur

[1] The VRML 2.0 Specification, WWW, 1996

[2] Conal Elliott, The Essence of ActiveX Animation, WWW, 1996

[3] Conal Elliott, A Brief Introduction to Active VRML, Technical Report MSR-TR-96-05, Microsoft Research, WWW, 1996

[4] Alligator Descartes, Interfacing Java and VRML, UK Java Developer's Conference Paper, WWW, 1996