VRML

Oliver Vornberger, Universität Osnabrück

VRML, sprich Wörmel, ist eine für das WWW entworfene *Virtual Reality Modelling Language* zur Beschreibung von 3-dimensionalen Szenen mit multimedialen Komponenten und Animation. Die gerenderte Projektion der Szene kann von jedem Web-Browser betrachtet werden, der über ein passendes Plugin verfügt. Dieser Artikel motiviert die Entwicklung von VRML und stellt beispielhaft die wichtigsten Sprachkonstrukte vor.

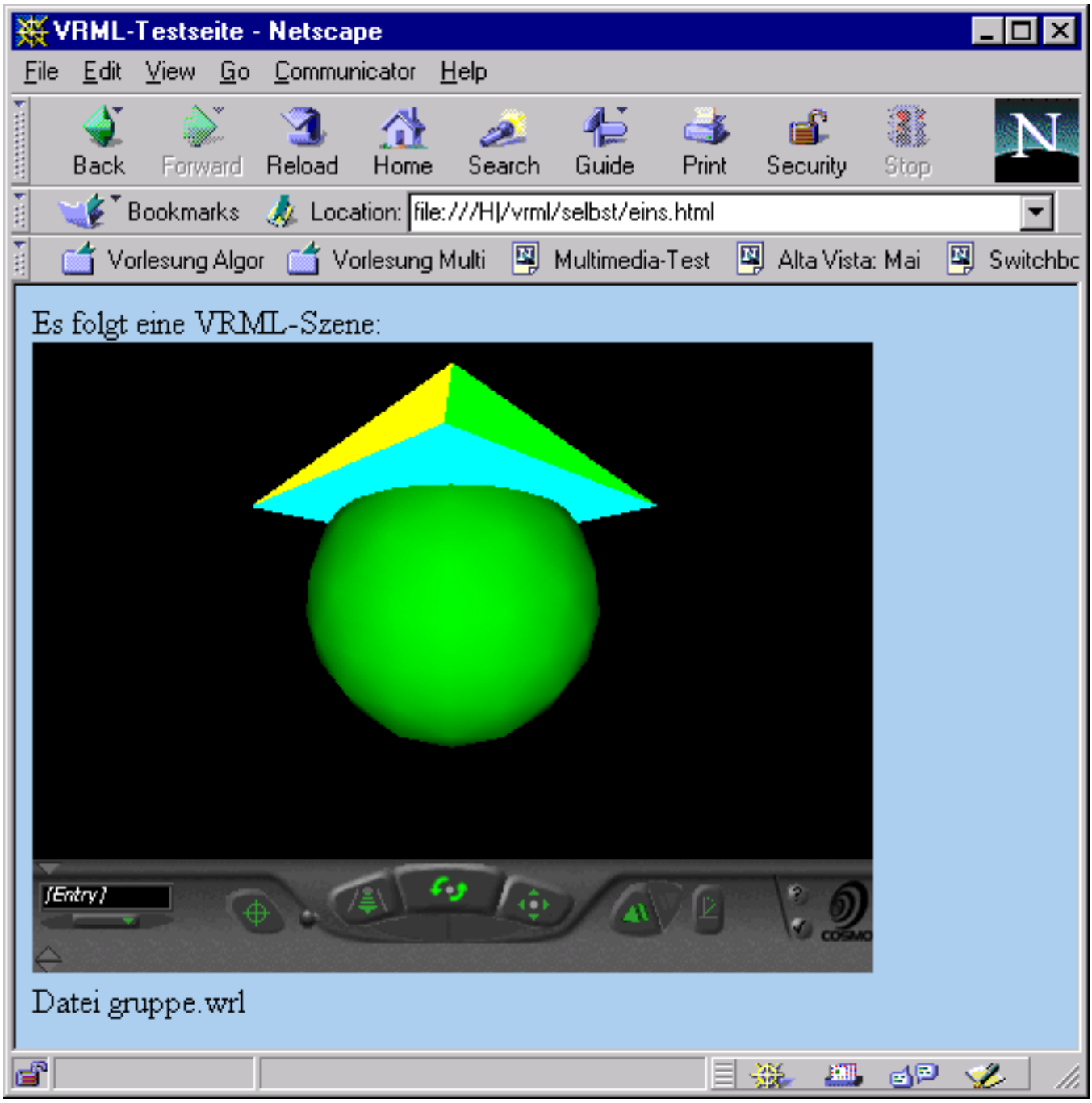

Abbildung 1: Screenshot vom Web-Browser mit VRML-Szene im Cosmo-Player

```
#VRML V2.0 utf8

Transform {                          # Plaziere
  translation 0 0 0                  # im Ursprung
  children [
    Shape {                          # eine Gestalt
      geometry Sphere {              # von der Form einer Kugel
        radius 1.5                   # mit Durchmesser 1.5
      }
      appearance Appearance {        # in der Erscheinung
        material Material {          # mit Materialbeschaffenheit
          diffuseColor 0 1 0         # gruene Farbe
          shininess 0.9              # stark reflektierend
        }
      }
    }
  ]
}
```

Abbildung 2: kugel.wrl: grüne, stark reflektierende Kugel mit Radius 1.5

```
#VRML V2.0 utf8

Shape {
  geometry IndexedFaceSet {

    coord Coordinate {
      point [            # beteiligte Punkte
         0  3  0         # 0. Pyramidenpunkt (Spitze)
         0  0 -2         # 1. Pyramidenpunkt (Norden)
        -2  0  0         # 2. Pyramidenpunkt (Westen)
         0  0  2         # 3. Pyramidenpunkt (Sueden)
         2  0  0         # 4. Pyramidenpunkt (Osten )
      ]
    }

    coordIndex [         # Polygone gegen Uhrzeiger, Ende: -1
         4  3  2  1 -1   # 0. Face: Punkte 1 2 3 4 (Grundflaeche)
         0  1  2    -1   # 1. Face: Punkte 0 1 2   (Nordwesten)
         0  2  3    -1   # 2. Face: Punkte 0 2 3   (Suedwesten)
         0  3  4    -1   # 3. Face: Punkte 0 3 4   (Suedosten)
         0  4  1         # 4. Face: Punkte 0 4 1   (Nordosten)
    ]

    colorPerVertex FALSE
    color Color {
        color [          # pro Face eine Farbe benennen
          0 1 1          # 0. Face: Cyan
          1 0 0          # 1. Face: Rot
          1 1 0          # 2. Face: Gelb
          0 1 0          # 3. Face: Gruen
          0 0 1          # 4. Face: Blau
        ]
    }
  }
}
```

Abbildung 3: pyramide.wrl: selbstdefinierte 5-seitige Pyramide

```
#VRML V2.0 utf8

Transform{                                  # Plaziere
  children[                                 # die folgenden Objekte:

    Anchor {                                # Hyperlink
      url "multimedia.wrl"                  # zur VRML-Datei multimedia.wrl
      description "Next world"              # Anzeige im Statusfenster
      children[                             # zugeordnet ist die Welt
        Inline {url "kugel.wrl"}            # aus Datei kugel.wrl
      ]
    }

    Transform {
      translation  0 1   0                  # hebe 1 Einheit nach oben
      scale        1 0.5 1                  # skaliere bzgl. y auf 50 %
      rotation     1 0   0 -0.523333        # kippe um 30 Grad nach hinten
      children[                             # verwende Welt
        Inline {url "pyramide.wrl"}         # aus Datei pyramide.wrl
      ]
    }

  ]
}
```

Abbildung 4: gruppe.wrl: Kugel mit Hyperlink unter gekippter Pyramide

```
#VRML V2.0 utf8

Shape {                                 # ein Gestaltknoten
  geometry Box {size 1 1 1}             # ein Quader der Kantenlaenge 1
  appearance Appearance {               # mit dem Aussehen
    texture ImageTexture {              # einer Bild-Textur
      url "posaune.jpg"                 # aus der JPEG-Datei posaune.jpg
    }
  }
}

Sound {                                 # ein Soundknoten
  source AudioClip {                    # gespeist von einem Audio-Clip
    url  "ragtime.mid"                  # aus der Midi-Datei ragtime.mid
    loop TRUE                           # in einer Endlosschleife
  }

  location  0 0 0                       # Schallquelle im Ursprung
  direction 0 0 1                       # dem Betrachter zugewandt
  minFront 1                            # Hoerbereichsanfang
  maxFront 8                            # Hoerbereichsende
}
```

Abbildung 5: multimedia.wrl: Quader mit Bild-Textur und Soundquelle

```
#VRML V2.0 utf8

Group {                                 # plaziere Gruppenknoten
  children [                            # bestehend aus
    DEF Taste TouchSensor {}            # einem Touch—Sensor
    Inline { url "kugel.wrl" }          # und einer Kugel
  ]
}

Sound {                                 # plaziere Soundknoten
  source DEF Tut AudioClip {            # gespeist von Audio—Clip
    url "tut.wav"                       # aus der Wave—Datei tut.wav
  }
  minFront  5                           # Anfang des Schallbereichs
  maxFront 50                           # Ende des Schallbereichs
}

ROUTE Taste.touchTime                   # bei Beruehrung der Kugel
   TO Tut.set_startTime                 # schicke Systemzeit an den Knoten Tut
```

Abbildung 6: interaktion.wrl: Kugel macht Geräusch bei Berührung

```
#VRML V2.0 utf8

DEF Schieber PlaneSensor {}             # Sensor zum Melden einer Mausbewegung

DEF Timer TimeSensor {                  # Sensor zum Melden eines Zeitintervalls
  cycleInterval 5                       # Dauer 5 Sekunden
  loop TRUE                             # Endlosschleife
}

DEF Rotierer OrientationInterpolator{   # Interpolator fuer Rotation
  key      [0 , 1]                      # bilde Schluessel 0 und 1 ab auf
  keyValue [ 0  1  0  0                 #   0 Grad Drehung bzgl. y
             0  1  0  3.14]             # 180 Grad Drehung bzgl. y
}

DEF Pyramide Transform {                # plaziere Objekt mit Namen Pyramide
      children [                        # bestehend aus
        Inline {url "pyramide.wrl"}     # VRML—Datei pyramide.wrl
      ]
}

ROUTE Timer.fraction_changed            # falls Zeitintervall sich aendert
   TO Rotierer.set_fraction             # schicke Bruchteil an Rotierer
ROUTE Rotierer.value_changed            # falls Drehung sich aendert
   TO Pyramide.set_rotation             # schicke Drehwert an Pyramide

ROUTE Schieber.translation_changed      # falls gedrueckter Mauszeiger bewegt wird
   TO Pyramide.set_translation          # schicke Translationswert an Pyramide
```

Abbildung 7: animation.wrl: selbständig sich drehende und interaktiv verschiebbare Pyramide

```
#VRML V2.0 utf8

Group {                                  # gruppiere
  children [                             # folgende Objekte
    DEF Taste TouchSensor{}              # Beruehrungssensor Taste
    DEF Pyramide Transform {             # Objekt Pyramide
      children[                          # bestehend aus
        Inline {url "pyramide.wrl"}      # VRML-Welt pyramide.wrl
      ]
    }
  ]
}

DEF Aktion Script {                      # Script mit Namen Aktion
  eventIn  SFBool     isActive           # Input-Parameter
  eventOut SFRotation drehung            # Output-Parameter
  url [                                  # gespeist
     "javascript:                       // von inline-Javascript
     function isActive(eventValue) {    // fuer den eventIn zustaendig
       if (eventValue == true) {        // falls isActive den Wert wahr hat
         drehung[0] += 0.3;             // drehe weiter bzgl. x-Achse
         drehung[1] += 0.2;             // drehe weiter bzgl. y-Achse
         drehung[2] += 0.1;             // drehe weiter bzgl. z-Achse
         drehung[3] += 0.174444;        // erhoehe Drehwinkel um 10 Grad
       }
     }"
  ]
}

ROUTE Taste.isActive                     # bei Beruehren der Pyramide
   TO Aktion.isActive                    # sende Nachricht an das Script Aktion

ROUTE Aktion.drehung                     # vom Script Aktion erzeugter Dreh-Vektor
   TO Pyramide.set_rotation              # wird an die Pyramide geschickt
```

Abbildung 8: javascript.wrl: Rotation eines Objekts wird über Javascript manipuliert

Motivation

Bei der Nutzung von Computer-Ressourcen läßt sich ein weiterer Paradigmenwechsel beobachten: Während in der EDV-Gründerzeit speziell ausgebildete Techniker am Mainframe-Computer Befehle eintippten, danach einige Jahrzehnte später Kopfarbeiter per *Drag & Drop* Fensterelemente manipulierten, surft inzwischen jedermann und -frau in einer weltweit vernetzten, multimedialen Landschaft. Der Kontext hat sich also von institutionell über persönlich zu sozial gewandelt.

Aus diesem Zeitgeist heraus diskutierten im April 1994 auf der 1st International WWW Conference in Genf Tim Berners-Lee, einer der Väter des WWW, mit den beiden Autoren des Systems Labyrinth, Mark Pesce und Tony Parisi, über die Möglichkeit, 3-dimensionale begehbare Welten im Internet verfügbar zu machen. Ziel sollte es sein, jedem Web-Surfer nach dem Download einer kompakten Szenenbeschreibung eine fortwährende fotorealistische Projektion zu verschaffen, wie sie sich aus seinem interaktiv wählbaren Betrachterstandpunkt ergeben würde. Als Anwendungen kommen damit alle Visualisierungsvorhaben im Bereich Architektur, Medizin und Tourismus in Frage, wo Anbieter und Nutzer räumlich getrennt sind und nun per Internet über eine standardisierte Schnittstelle das Eintauchen in virtuelle Welten möglich wird.

Es wurde eine Mailing List aufgesetzt, die schon nach wenigen Wochen mit mehr als 1000 Teilnehmern über Syntax für Strukturen, Verhalten und Kommunikation debattierte. Bereits im Oktober 1994 wurde auf der 2nd International WWW Conference in Chicago VRML 1.0 vorgestellt, ein Entwurf, der wesentlich vom Silicon Graphics System *Open Inventor* inspiriert war. VRML 1.0 konnte bereits geometrische Grundkörper und Polygone in einem Koordinatensystem plazieren und ihre Farbe und Materialeigenschaften spezifizieren. Auch ließen sich durch Hyperlinks Objekte beliebig im Web referieren. Abgesehen von dieser Möglichkeit der Interaktion handelte es sich allerdings um rein statische Szenen.

Diesem Manko sollte eine schnellstens eingerichtete *VRML Architecture Group* (VAG) abhelfen, welche Überlegungen zur Animation und zur Integration multimedialer Komponenten wie Sound und Video koordinierte. Zur 1st International VRML Conference im Dez. 1995 war es dann soweit: Als Sieger einer Ausschreibung für VRML 97 ging *Moving Worlds* von Silicon Graphics nach einer On-Line-Wahl mit 70 % der abgegebenen Stimmen durchs Ziel. Eine überarbeitete Syntax sorgte für die Beschreibung statischer Szenen mit multimedialen Bestandteilen, und ein neues Event-Handling-Konzept erlaubte Animation dieser Szenen sowie Interaktion mit dem Benutzer.

Syntax

VRML-Szenen werden beschrieben in ASCII-Dateien mit der Endung ***.wrl**, welche innerhalb einer HTML-Seite mit dem **EMBED**-Kommando referiert werden, z.B.

```
<EMBED SRC="kugel.wrl"
WIDTH=400 HEIGTH=300>
```

Ein entsprechend konfigurierter Web-Server schickt dem anfordernden Klienten als Vorspann dieser Daten den *Mime*-Typ VRML, worauf das zur Betrachtung installierte Plugin, z.B. *Cosmo Player* 2.0 von Silicon Graphics, die eingehenden Daten in eine interne Datenstruktur einliest, von wo sie zur Berechnung einer fotorealistischen Projektion immer wieder ausgelesen werden. Abbildung 1 zeigt einen Screenshot.

In welcher Weise Blickwinkel und Orientierung in der Szene modifiziert werden können, bleibt dem Plugin überlassen: Mit Mauszeiger und Keyboard-Shortcuts wandert der Benutzer durch eine virtuelle Welt, verkörpert im wahrsten Sinne des Wortes durch einen Avatar, seiner geometrischen Repräsentation, beschränkt in seiner Beweglichkeit durch physikalische Restriktionen und durch eine simulierte Schwerkraft.

Geometrie

VRML-Szenen sind hierarchisch aufgebaut und durch Klammerpaare strukturiert. Wichtigster Bestandteil ist der sogenannte *Knoten* (meistens mit großem Anfangsbuchstaben geschrieben), der ähnlich dem Record einer Programmiersprache aus Feldern verschiedenen Typs besteht (meistens klein geschrieben). Diese Felder verweisen entweder auf nicht weiter strukturierte Objektknoten oder andere Gruppenknoten, die wiederum mittels ihrer Felder verzweigen können.

Abbildung 2 zeigt den Aufbau einer Szene, in der eine Kugel mit Radius 1.5 im Ursprung des Weltkoordinatensystems plaziert wird. Die x-Richtung entspricht der horizontalen Bewegung, y beschreibt die vertikale Richtung, und z wächst auf den Betrachter zu. Der *Sphere*-Knoten hat dabei als einziges (optionales) Feld den Radius. Diese Kugel wird referiert über das *geometry*-Feld des *Shape*-Knotens, zuständig für die Gestaltung eines Objekts. Über das *appearance*-Feld wird die Materialbeschaffenheit in Form einer RGB-Farbe und eines Transparenz-Koeffizienten spezifiziert. Der *Shape*-Knoten wiederum ist als eins der Kinder im Transform-Knoten eingetragen, der über das *translation*-Feld für die Verschiebung der Kugel sorgt.

Neben den Grundbausteinen *Sphere* (Kugel), *Box* (Quader), *Cone* (Kegel) und *Cylinder* (Zylinder) lassen sich eigene geometrische Gebilde konstruieren. Bei einem *IndexedFaceSet* werden zunächst alle Ecken als Punkte mit 3D-Koordinaten definiert (*coord* in Abbildung 3). Die Oberfläche des Körpers wird dann als Menge von Seiten beschrieben, die als Polygone mit diesen Punkten angegeben werden (*coordIndex*). Abbildung 3 zeigt die Definition einer 5-farbigen Pyramide mit quadratischer Grundfläche.

Wiederverwendung

Zur Reduktion der Dateigrößen und zur besseren Lesbarkeit lassen sich einmal spezifizierte Welten wiederverwenden. Abbildung 4 zeigt die Kombination der beiden graphischen Objekte Kugel und Pyramide, wobei die Pyramide leicht nach hinten gekippt oberhalb der Kugel positioniert wird. Ferner wurde ein Hyperlink eingerichtet, der zu einer weiteren VRML-Welt führt, sobald der Mauszeiger auf der Kugel gedrückt wird.

Multimedia

Bilder, Sound und Video sind wichtige Bestandteile von VRML-Welten. Mit wenigen Vokabeln lassen sich solche multimedialen Komponenten in die Szene integrieren. Dabei bieten sich zum Einfärben von Oberflächen 2-dimensionale Pixel-Bilder im JPEG-Format an oder auch Quicktime-Filme und MPEG-Dateien. Sound-Knoten werden gespeist von *Wave*-Dateien oder auch *Midi*-Files, welche wegen ihres kompakten Speicherformats besonders zum Verschicken im WWW geeignet sind. Abbildung 5 zeigt einen Quader, dessen 6 Flächen mit *Texture-Mapping* perspektivisch gerendert werden. In seinem Zentrum wurde eine Schallquelle positioniert, welche bei Annäherung durch den Benutzer von vorne schrittweise lauter zu hören ist.

Interaktion

VRML97 bietet zahlreiche Möglichkeiten, mit denen einer Szene Dynamik und Interaktion verliehen werden kann. Die zentralen Bausteine für die hierzu erforder-

liche Ereignisbehandlung sind die *EventIn*- bzw. *EventOut*-Felder von Knoten, mit denen Meldungen empfangen und Zustandsänderungen weitergeschickt werden können. Es gibt *Time*-, *Proximity*-, *Visibility*-, *Collision*- und *Touch*-Sensoren, welche das Verstreichen einer Zeitspanne, das Annähern des Benutzers, die Sichtbarkeit von Objekten, das Zusammentreffen des Avatars mit einem Objekt und die Berührung mit dem Mauszeiger signalisieren. Verständlicherweise müssen Typ des verschickenden Ereignisfeldes und Typ des empfangenden Ereignisfeldes übereinstimmen.

Abbildung 6 zeigt die Kugel, versehen mit einem *Touch*-Sensor, welcher bei Mausdruck eine Nachricht an den *Sound*-Knoten schickt, der auf diese Weise seinen Spielbeginnzeitpunkt erhält und die zugeordnete *Wave*-Datei startet. Mit der Vokabel **DEF** werden Namen für Objekte hinterlegt, auf die dann Event-Verbindungen mit **ROUTE** Bezug nehmen können.

Animation

Objekte können sich in VRML bewegen, entweder selbständig oder weil der Benutzer dies zum Beispiel mit der Maus veranlaßt. Organisiert werden solche Bewegungen wiederum mit Hilfe der Ereignisbehandlung. In Abbildung 7 kann man ein Objekt mit der Maus verschieben, welches sich außerdem selbständig dreht. Genauer: Die Ziehbewegung des gedrückten Mauszeigers wird zur Manipulation der lokalen Translation eines Objekts verwendet, und das regelmäßige Verstreichen eines Zeitintervalls löst eine Nachricht an denselben geometrischen Knoten aus, der hierdurch seinen aktuellen Rotationsvektor erhält. Eine solche Konstruktion verlangt einen *Orientation-Interpolator*, welcher beliebige Bruchzahlen zwischen 0 und 1 auf die zugeordneten Werte seines Schlüsselintervalls abbildet, hier bestehend aus allen Drehwinkeln zwischen 0 und 3.14 (= 180 Grad beschrieben in Bogenmaß), bezogen auf die y-Achse.

Scripts

Manchmal reichen die in VRML angebotenen Funktionen wie Sensoren und Interpolatoren nicht aus, um ein spezielles situationsbedingtes Interaktionsverhalten zu erzeugen. Abhilfe schafft hier der sogenannte *Script*-Knoten, welcher Input empfangen, Daten verarbeiten und Output verschicken kann. Beispielsweise kann eine vom *Touch*-Sensor geschickte Nachricht eine Berechnung anstoßen, deren Ergebnis in Form einer Translations-Nachricht an ein bestimmtes Objekt geschickt und dort zur Neupositionierung genutzt wird.

Die Formulierung des Berechnungsalgorithmus geschieht entweder durch ein *Javascript*-Programm, inline gelistet im Script-Knoten, oder durch eine assoziierte *Java*-Klasse, welche in übersetzter Form mit Dateiendung ***.class** lokal im Verzeichnis oder entfernt im Netz liegt. Zum Übersetzen der *Java*-Quelle ist das *External Authoring Interface* (EAI) erforderlich, welches in Form einiger Packages aus dem Verzeichnis importiert wird, in welches sie das VRML-Plugin bei der Installation deponiert hatte.

Abbildung 8 zeigt die Pyramide, zusammen mit einer *JavaScript*-Funktion, die bei jedem Aufruf den Drehwinkel bzgl. der y-Achse um weitere 10 Grad erhöht.

Multiuser

Eines der ursprünglichen Entwicklungsziele von VRML bleibt auch bei VRML97 offen: Es gibt noch keinen Standard für Multiuser-Welten. Hiermit sind Szenen gemeint, in denen mehrere Benutzer gleichzeitig herumwandern und interagieren können. Da jedes ausgelöste Ereignis von allen Beteiligten wahrgenommen werden soll, muß ein zentraler Server den jeweiligen Zustand der Welt verwalten und den anfragenden Klienten fortwährend Updates schicken. In diesem Zusammenhang erhält der Avatar eine aufgewertete Rolle: Zusätzlich zu seiner geometrischen Räumlichkeit, die schon zur Kollisionsdetektion in einer Single-User-Szenerie benötigt wurde, muß nun auch sein visuelles Äußeres spezifiziert werden, sicherlich ein weiterer wichtiger Schritt zur Verschmelzung eines real existierenden Benutzers mit der von ihm gespielten Rolle. Spätestens hier wird klar, daß Navigation in einer VRML-Welt weit mehr bedeutet als 3D im Internet.

Literatur

[1] J. Hartman, J. Wernecke *The VRML 2.0 Handbook — Building Moving Worlds on the Web* Addison-Wesley, 1996.

[2] J. Kloss, R. Rockwell, K. Szabo, M. Duchow *VRML97 — Der neue Standard für interaktive 3D-Welten im World Wide Web* Addison-Wesley, 1998. □