Szenengraph-Architekturen im Kontext von VR- und AR-Anwendungen

Hauptseminar Medieninformatik

Christina Eicher

10. Mai 2004

Inhalt

Szene

Eine Szene umfasst sämtliche für das Rendering benötigten Informationen, z. B.

- geometrische Beschreibungen der Visualisierungsobjekte
- visuelle Attribute (z. B. Farben)
- Beleuchtungssituation
- Kameraeinstellungen

➔ Eine Szene ist nicht als statische Einheit zu betrachten!

Szenengraphen

- gerichteter, azyklischer Graph ("directed acyclic graph", kurz DAG)
- Die Knoten des Szenengraphen sind mit Rendering-Objekten verknüpft und organisieren diese hierarchisch
- Mit dem Szenengraphen zu verknüpfen sind eine Kamera und eine Beleuchtungssituation

➔ Häufig wird von Baum-Strukturen statt DAGs gesprochen.

Vorteile

- Möglichkeit einer einfachen Attributspezifikation, die sich auf mehrere Teilgraphen beziehen soll, z.B.
  - → Festlegung eines gemeinsamen Erscheinungsbildes für mehrere Visualisierungsobjekte
- Vereinfachung der Konstruktion komplexer Objekte
- bessere Speicher-Ausnutzung
- einfachere Bild-Aktualisierung bei Objekt-Änderungen

Rendering-Objekte (1) - Grafische Attribute

- steuern die visuelle Erscheinung ("appearance") der Shapes

Beispiele:

→ Angaben zur Oberflächencharakteristik (Farben, Texturen, Reflektionskoeffizienten, Lichtemissionseigenschaften, etc.)

→ Angaben zur Repräsentationsqualität (Tesselationstiefe, Shading-Verfahren, Linienbreiten in Pixeln etc.)

Rendering-Objekte (1) - Geometrische Attribute

- Manipulation der Objektgeometrien

Beispiele für geometrische Transformationen:

- Translationen
- Skalierungen
- Rotationen
- Objekte, welche die Blick- oder Bewegungsrichtung der Kamera liefern

Verhaltensklassen

Die heute verfügbaren Szenengraph-Bibliotheken stellen eine Vielzahl verschiedener Verhaltensklassen zur Verfügung. Zu den wichtigsten Klassen zählen:

- Klassen zur Unterstützung der Kamerasteuerung
- Klassen zur Handhabung Benutzer-generierter Ereignisse
- Picking-Klassen, welche die Ermittlung eines im Bild angeklickten Shapes ermöglichen
- Level-of-Detail-Klassen
- Klassen zur Umsetzung des sogenannten Billboard-Verhaltens, durch welches sich Shapes in Richtung des Betrachters ausrichten lassen
- Klassen zur Kollisionserkennung
- Klassen zur Unterstützung der Spezifikation zeitlicher Abläufe
- Klassen zur Unterstützung von Movietexturen

➔ Nicht jedes API bietet alle genannten Verhaltensklassen an!!

Anwendungsbereich

- Beschreibung dreidimensionaler Szenen im Zusammenhang mit VR- und AR-Anwendungen

Szenengraphen werden dabei hauptsächlich benötigt für:

- Rendering
- Definition und Verwaltung komplexer Visualisierungsobjekte
- interaktives und zeitliches Verhalten einer Szene

Wichtige Szenengraphen

Wichtige Bibliotheken, welche die Anordnung von Objekten in Szenengraphen ermöglichen:

- Open Inventor
- Coin3D
- OpenGL Performer
- Open Scene Graph
- Open SG
- VRML/X3D
- Java3D

Open Inventor

Allgemeines:

- Objekt-orientiert programmiert (C++)
- Basiert auf OpenGL
- Graphik-Zugriff über Scene-Database
- Interaktion/Animation mittels Manipulatoren und Engines
- File-Format: iv-Format
- bietet Szenenoptimierung und Komponenten wie Viewer, Editoren, etc.

Nodes

- Elemente des Szenengraphen
- die zugehörigen Klassen beginnen mit ‘So‘ (von Scene-Object)
- Jeder Knoten besitzt eine Menge von Feldern

Fields:

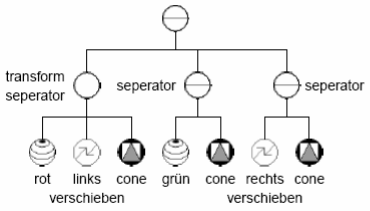
- beinhalten Informationen, die den Knoten charakterisieren (Parameter)
- Klasse der Scene-Basic-Objekte (beginnen mit ‘Sb‘)
- können mit Feldern anderer Knoten verknüpft werden

Grundlegende Node-Klassen:

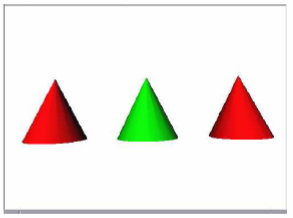
- shape nodes: Primitive
- property nodes: Eigenschaften wie Licht und Material
- group nodes: Container, die Knoten in Untergraphen zusammenfassen

Traversierung und Seperatoren

Beispiel: Einfluß von Seperatoren



Scene-Graph

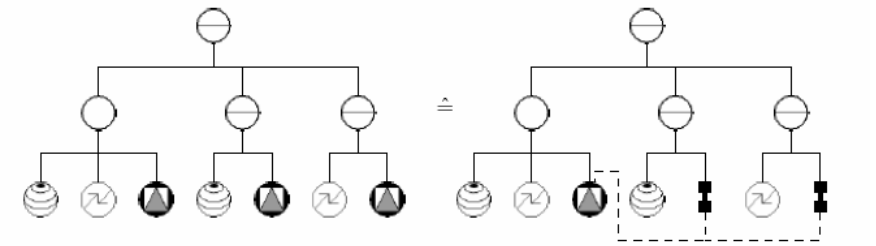


Szene

Bevor die Kinder eines SoSeperator durchlaufen werden, wird der aktuelle Zustand („traversal state“) gespeichert; nach dem Durchlaufen aller Kinder wird der ursprüngliche Zustand wieder hergestellt (→ Kapselung)

- das rote Material wirkt sich global aus, das grüne Material nur lokal
- alle Transformationen wirken sich lokal aus

Referenzen



Licht und Kamera

Camera:

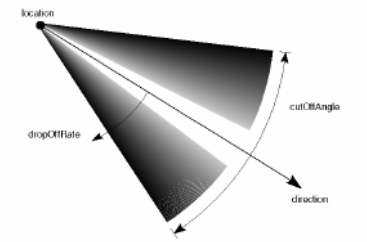
- Kamera muß vor allen Objekten der Szene im Graph stehen

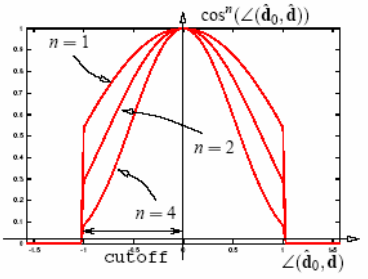
Lichtquellen:

- beleuchten nur nachfolgende Objekte im Graph
- Position und Orientierung entsprechend der aktuellen Transformationsmatrix

Lichtarten:

- Punktlicht
- Richtungslicht/paralleles Licht
- Punktlicht mit Hauptausstrahlrichtung





Intensitätsverteilung Spot-Light

Aktionen

- Actions werden direkt auf den Szenengraph (oder Teil-Graph) angewendet
- Action wird für jeden Node ausgeführt

Beispiele für Actions:

- SoGLRenderAction: Darstellen der Szene entsprechend der Kamera
- SoWriteAction: Scene-Graph in Datei schreiben
- SoSearchAction: Sucht Pfade zu spezifischen Nodes

Sensoren

- Reaktion auf Zustandsänderungen im Scene-Graph
- Reaktion auf Zeit-Ereignisse
- Implementierung von Animationen

Daten-Sensoren

- sind einem Field, einem Node oder einem Pfad zugeordnet
- triggern, sobald sich ein Datum geändert hat

Zeit-Sensoren:

- SoAlarmSensor: Einmaliges Triggern zu festgelegtem Zeitpunkt
- SoTimerSensor: Triggern in vorgegebenen Abständen, z.B. für Animationen

→ Sensoren sind keine Nodes!

Speichermanagement

- OpenInventor hat ein eigenes Speicher-Management
- Referenz-Zähler: Jeder Node zählt Referenzen zu anderen Nodes
- Initial: Ref-Count auf 0
- Einhängen in die Hierarchie, Löschen von Kindern etc.

  → geänderter Ref-Count
- Ref-Count auf 0 reduziert → Node-Objekt wird gelöscht

Der Astronauten-Trainingssimulator

Beteiligte Institutionen:

- Johnson Space Center in Houston
- University of Houston Downtown
- Fraunhofer-IGD

zu simulierende Mission:

- Austauschen eines defekten Aggregats des Hubble Space Telescopes

Technische Voraussetzungen

- zwei gleichzeitig laufende VRSysteme
- grafisches Modell in Form von Inventor-Dateien
- ISDN-Standleitung zur Synchronisation
- Sprechverbindung über ein analoges Telefon.
- Head Mounted Display (Monitorbrille) mit Headtracker
- Datenhandschuh mit integriertem Tracker

Modell

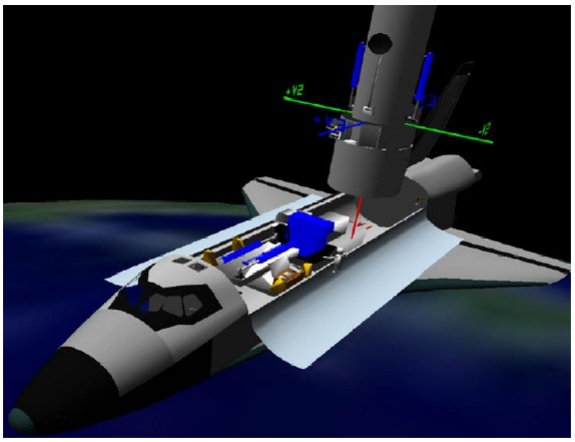


Abbildung 6.1: Modell des Space Shuttle mit angedocktem Hubble Teleskop

Szenengraph

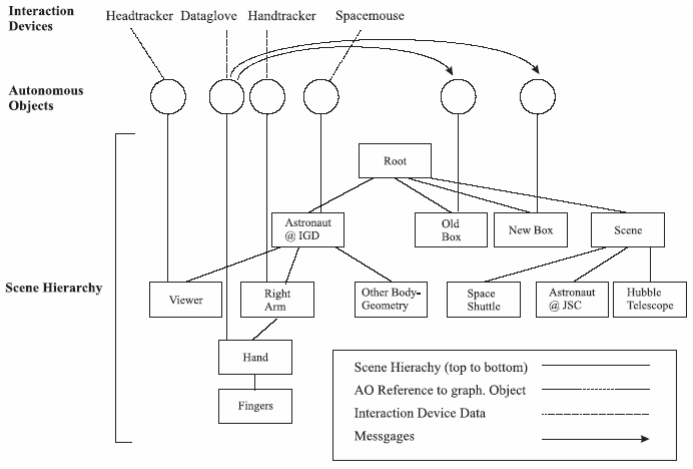


Abbildung 6.2: Autonome Objekte im Astronauten Trainingssimulator

Durchführung des Experiments

- wichtige Operationen: Greifen und Loslassen von Objekten
- realisiert durch Kollisions- und Gestenerkennung
- Änderungen im Szenengraphen notwendig
- Aktualisierung der Transformation eines grafischen Objektes, das sich bewegt hat, nötig
- Experiment wurde positiv bewertet

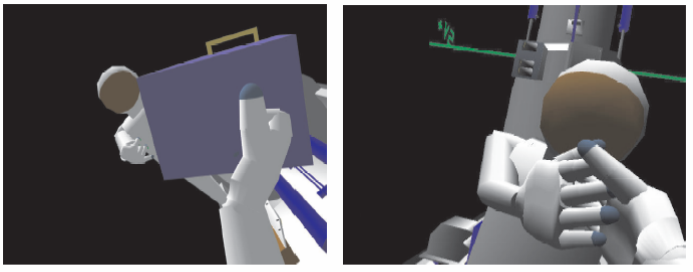


Abbildung 6.3: Szene aus Sicht des europäischen Astronauten

Quellen

Allgemein:
http://ifgivor.uni-muenster.de/vorlesungen/3D_geovisualisierung/NeueMedien2_16.htm
http://www.gris.uni-tuebingen.de/projects/vis/coursebook/rendering/engine/
http--medien.informatik.uni-ulm.de-lehre-courses-ss02-Computergrafik-DetlefKoentges.pdf

Open Inventor:
http://www.cg.tuwien.ac.at/studentwork/VRSem96/Inventor/
http://www.ifmi.org/old/full/education/ss2002/vmp/EinfuehrungsBlatt.pdf
http://www.vis.uni-stuttgart.de/img/sommer/seminar/Vortrag-OpenInventor.pdf
http://www.umi.cs.tu-bs.de/full/education/practical/ss2003/vmp2003/EinfuehrungsBlatt.pdf
http://www.cg.fb12.uni-siegen.de/Lehre/CG2/Vorlesung%20Material/cg2_12.pdf

Anwendungsbeispiel:
http--elib.tu-darmstadt.de-diss-000262-6-conclusion.pdf