Hauptseminar „Virtual und Augmented Reality“
Prof. Dr. Heinrich Hussmann
Betreuung: Dipl.-Inf. Arnd Vitzthum
Lehrstuhl für Medieninformatik
Ludwig-Maximilians-Universität München
Sommersemester 2004

Seminararbeit

Szenengrafarchitekturen im Kontext von VR- und ARAnwendungen

Christina Eicher

Abstract
Virtual und Augmented Reality-Anwendungen gewinnen im Laufe der Zeit zunehmend an Bedeutung. Nicht nur in der Spieleindustrie bedient man sich immer häufiger möglichst realitätsnahen 3D-Szenarien, auch die Forschung, die Medizin und das Militär nutzen die Möglichkeiten der Simulation von dreidimensionalen Umgebungen für ihre Zwecke. In diesem Zusammenhang wachsen auch die Anforderungen an die Systeme. Es wird zunehmend mehr Wert auf Präzision und Schnelligkeit der Anwendungen gelegt. Um dies zu erreichen, ist eine gute Organisation der 3D-Objekte notwendig, die auch eine Einbindung der virtuellen in die reale Welt ermöglicht. Eine nicht unwesentliche Rolle spielen hierbei die Szenengraphen, welche eine klar strukturierte Verwaltung aller beteiligten Objekte ermöglichen. Im Folgenden werden Funktion und Leistungsumfang dieser Architekturen etwas genauer erläutert werden.

Vorwort
Der erste Teil der Arbeit beschäftigt sich mit den grundlegenden Eigenschaften und Leistungsmerkmalen von Szenengraphen. Da der Schwerpunkt dieser Arbeit allerdings nicht auf eine theoretische Abhandlung über Szenengraph-Architekturen abzielt, sondern vielmehr auf eine praxisorientierte Einführung in die Welt der Szenengraphen, wird dieser Teil bewusst nur kurz angeschnitten, um die Grundlage für den zweiten Teil zu bilden. Selbiger gibt einen Überblick über die auf OpenGL basierende Bibliothek Open Inventor von Silicon Graphics. Nachdem der Leser nun die Grundlagen von Szenengraphen kennengelernt hat, wird die Arbeit mit einem Beispiel aus der Praxis abschließen, für das der Astronauten-Trainingssimulator des Johnson Space Center in Houston gewählt wurde.

1. Szenengraph-Architekturen

1.1. Szenengraphen

Um eine komplexe 3D-Szene mit sämtlichen für das Rendering benötigten Informationen adäquat verwalten zu können, benötigt man eine gut organisierte Struktur, die gleichzeitig ein hohes Maß an Flexibilität bietet. Es reicht im Allgemeinen nicht aus, die geometrische Beschreibung und die visuellen Attribute sämtlicher Visualisierungsobjekte, einschließlich der jeweiligen Beleuchtungssituation und der Kameraeinstellungen zu erfassen, obwohl dies allein im Zusammenhang mit umfangreichen 3D-Szenarien bereits eine nicht zu unterschätzende Herausforderung darstellt. Man muss zudem beachten, dass es sich gerade bei VR- und AR-Anwendungen in der Regel nicht um statische Einheiten handelt. Vor allem aufgrund von Animationen und Interaktionen ergeben sich häufig dynamische Änderungen in einer dreidimensionalen Szene. Um dies alles erfassen und verwalten zu können, bedient man sich heutzutage in der Regel einer Szenengraph-Architektur.

Bei einem Szenengraphen handelt es sich um einen gerichteten, azyklischen Graphen, dessen Knoten mit Rendering-Objekten verknüpft sind. Diese sind baumartig im Graphen angeordnet. Aufgrund dieser hierarchischen Struktur ist es möglich, genau festzulegen, für welche Teilgraphen eine bestimmte Attributspezifikation gelten soll. Da sich eine bestimmte Information normalerweise auf sämtliche Folgeknoten – und nur auf diese – auswirkt, kann man zum Beispiel ein gemeinsames Erscheinungsbild für mehrere Visualisierungsobjekte gleichzeitig festlegen. So ist es unter anderem sinnvoll, Informationen über Beleuchtung und Kameraeinstellungen in unmittelbarer Nähe der Wurzel des Graphen zu positionieren, wenn man die komplette Szene gerendert haben möchte, da sämtliche Knoten, die in der Hierarchie zuvor durchlaufen werden, nicht erfasst werden. Dieser Sachverhalt wird in Kapitel 2.3 noch einmal anhand eines Beispiels erläutert werden.

Eine wichtige Rolle bei der Beschreibung von Rendering-Objekten stellen grafische Attribute dar. Diese sind für das visuelle Erscheinungsbild der Objekte zuständig. Dazu gehören zunächst einmal Angaben zur Oberflächencharakteristik wie beispielsweise Farben, Texturen, Reflektionskoeffizienten oder Lichtemissionseigenschaften. Ebenfalls wichtig sind Informationen über Tesselationstiefe, Shading-Verfahren, Linienbreiten und dergleichen. Ein weiterer essentieller Aspekt in 3D-Szenen ist die Beschreibung der Objektgeometrien, ebenso wie die Manipulation selbiger. Diese kann unter anderem durch Translationen, Skalierungen und Rotationen bewirkt werden.

Sämtliche soeben erwähnten Informationen über die Objekte einer 3D-Anwendung werden durch die hierarchische und zugleich dynamische Struktur des Szenengraphen erfasst.

Obwohl sich die verschiedenen Szenengraph-Bibliotheken in ihrer Struktur und ihrem Leistungsumfang teilweise deutlich unterscheiden, gibt es einige wichtige Verhaltensklassen, die nahezu immer enthalten sind. In [1] werden einige dieser Klassen genannt, die allerdings nicht verbindlich in jeder Architektur vorhanden sein müssen. Im großen und ganzen muss man sich vor der Wahl einer bestimmten Szenengraph-Architektur darüber informieren, welche Funktionen im Leistungsumfang für die aktuelle Anwendung enthalten sein müssen und welche Bibliothek in diesem Fall am geeignetsten erscheint.

2. Open Inventor

2.1. Allgemeines

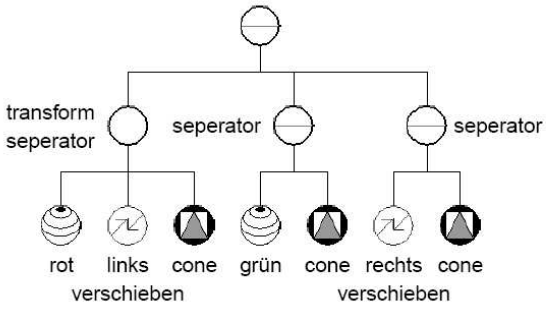
Nachdem soeben die allgemeine Struktur von Szenengrapharchitekturen kurz erläutert wurde, werden nun die theoretischen Grundlagen anhand eines Beispiels für eine Szenengraph-API etwas veranschaulicht. Hierfür wurde ein auf OpenGL basierender Standard gewählt: Open Inventor von Silicon Graphics. Im Gegensatz zu OpenGL ist Open Inventor allerdings nicht imperativ sondern objektorientiert in C++ programmiert, was unter anderem die Entwicklung objektorientierter Grafikprogramme ermöglicht. Ein weiterer Vorteil von OpenInventor ist zudem, dass ein eigenes File-Format (.iv-Format) existiert, welches den Austausch von Daten erheblich vereinfacht. Die Wichtigkeit dieses Aspekts wird auch im Zusammenhang mit dem Anwendungsbeispiel aus Teil 3 dieser Arbeit noch einmal deutlich werden, doch zunächst zu den Grundlagen von Open Inventor.

2.2. Nodes und Fields

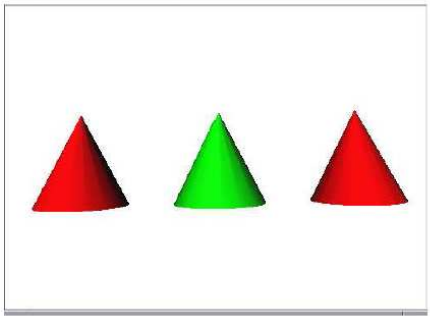
Die Knoten (Nodes) sind die wesentlichen Elemente des Szenengraphen. Bei Open Inventor beginnen die zugehörigen Klassen grundsätzlich mit dem Präfix ’So’, was für ’Scene-Object’ steht, also z.B. SoMaterial oder SoCone.
Jeder dieser Nodes besitzt eine Menge von Feldern (Fields), die Informationen beinhalten, welche den Knoten charakterisieren. Diese enthalten beispielsweise Parameter, welche die Größe, den Radius oder die Position eines Objektes beinhalten. Außerdem können Verknüpfungen zwischen den Feldern verschiedener Knoten bestehen. Die Field-Klassen sind sogenannte Scene-Basic-Objekte (beginnen immer mit ’Sb’). Zu diesen gehören unter anderen Matrizen, Vektoren, Ebenen oder Projektionen.
Es gibt nun einige grundlegende Node-Klassen. Hierzu gehören zunächst einmal die Shape-Nodes, welche einfache Primitive wie SoCone, SoCube oder SoSphere beinhalten. Des weiteren gibt es die Property-Nodes, deren Inhalt Attribute für Objekte sind. Dies können entweder Transformationen (Translation, Skalierung, Rotation), Angaben zur Geometrie (Vertex-Koordinaten etc.) oder Appearance-Informationen (Materialparameter, Komplexität, Detailgrad etc.) sein. Mit Hilfe von Group-Nodes lassen sich Knoten zu Sub-Graphen zusammenfassen. Außerdem gibt es noch eigene Knoten für Licht und Kameras, worauf in Kapitel 2.4 noch einmal näher eingegangen werden wird.

2.3. Traversierung und Seperatoren

In der Regel wird ein Szenengraph chronologisch von oben nach unten und von links nach rechts durchlaufen. Dabei gilt eine Information solange für alle nachfolgenden Objekte, bis sie durch eine neue Information überschrieben wird. Wird allerdings ein SoSeperator-Node durchlaufen, wird der aktuelle Zustand („traversal state“) gespeichert, hierauf werden sämtliche Kindknoten ganz normal durchlaufen und anschließend wird der gespeicherte Zustand wiederhergestellt. Dies führt zu einer Art Kapselung. Ein aus [7] entnommenes Beispiel soll das noch einmal kurz veranschaulichen:



Scene-Graph



Szene

Während sich die Informationen des linken Teilgraphen global auswirken, wirkt sich beispielsweise die Information, dass die Materialfarbe von rot zu grün wechselt, aufgrund des Seperators nur lokal aus, weshalb der rechte Kegel wiederum rot gefärbt ist.
Einen Sonderfall in Szenengraphen stellen die Transformationen dar, da sich selbige grundsätzlich nur lokal auswirken.
An dieser Stelle sei noch der sogenannte Switch Node erwähnt. Dieser bewirkt, dass von den Nachfolgeknoten immer nur einer zur Bearbeitung ausgewählt wird, was beispielsweise für Level-of-Detail-Darstellungen eines Objekts benötigt wird. Diese sind unter anderem dann relevant, wenn man ein Objekt aufgrund von unterschiedlichen Entfernungen vom Betrachter in verschieden genauen Auflösungen darstellen können muss.

2.4. Licht und Kamera

Wie bereits unter 2.2 erwähnt, stellt Open Inventor eigene Nodes für Beleuchtung und Kamera zur Verfügung. Aus diesem Grund ist es auch hier relevant, die Objekte an der richtigen Stelle im Scene-Graph zu positionieren, da die Kamera nur diejenigen Objekte erfasst, die in der Hierarchie weiter unten angesiedelt sind als sie selbst. Das selbe gilt für sämtliche Lichtquellen, die sich zusätzlich noch an der aktuellen Transformationsmatrix orientieren. Darauf soll allerdings an dieser Stelle nicht detaillierter eingangen werden, da das im Rahmen dieser Arbeit zu weit führen würde. Es sei lediglich noch kurz erwähnt, dass Open Inventor verschiedene Lichtarten bereitstellt, zu denen unter anderem Punktlicht, paralleles bzw. Richtungslicht und Punktlichter mit Hauptausstrahlrichtung gehören.

2.5. Aktionen

Nach der Erstellung des Szenengraphen, gibt es verschiedene Operationen, die auf ihn angewendet werden können. Man kann beispielsweise verschiedene Aktionen durchführen, die sich grundsätzlich auf den ganzen Szenengraphen oder zumindest auf einen gesamten Teilgraphen auswirken, wobei die Action für jeden Node ausgeführt wird. Ein Beispiel hierfür ist unter anderem die SoGLRenderAction, welche die Szene entsprechend der aktuellen Kameraeinstellung und den gesetzten Lichtquellen darstellt, oder die SoWriteAction, die dafür sorgt, dass der Scene-Graph in eine Datei geschrieben wird. Zuletzt sei noch die SoSearchAction erwähnt, die den Pfad zu einem spezifischen Node sucht, was durchaus in vielen Anwendungen sehr hilfreich sein kann.

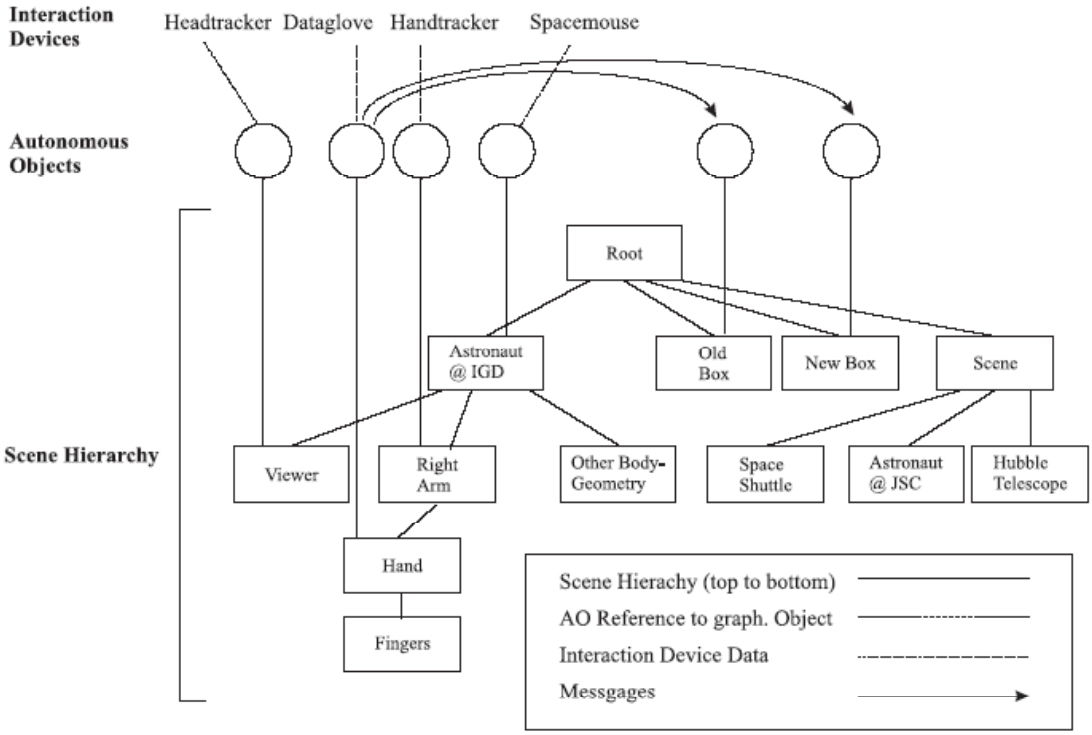
2.6. Sensoren

Open Inventor bietet dem Benutzer zwei Arten von Sensoren an. Dies sind zum einen die sogenannten Daten-Sensoren, die einem Field, einem Node, oder sogar einem ganzen Pfad zugeordnet sind und triggern[1], sobald sich ein Datum des zugehörigen Elements ändert. Es ist beispielweise äußerst sinnvoll, den Root-Knoten mir einem Sensor zu versehen, der bei jeder Änderung im Graphen ein Repaint der Szenen veranlasst.
Die zweite Variante sind die Zeit-Sensoren. Dazu gehört einerseits der SoAlarmSensor, der einmalig zu einem festgelegten Zeitpunkt triggert, und andererseits der SoTimerSensor, welcher in vorgegebenen Zeitabständen triggert. Dies kann beispielsweise für Animationen sehr interessant sein. Es sei an dieser Stelle noch erwähnt, dass es sich bei Sensoren nicht um eigenständige Nodes handelt.
Nachdem nun die Funktionsweise einer Szenengraph-Architektur etwas genauer beschrieben wurde, folgt nun ein konkretes Beispiel für den Einsatz eines Szenengraphen.

3. Der Astronauten-Trainingssimulator

Als Beispiel für die Anwendung einer Szenengraph-Architektur in einer Virtual-Reality-Umgebung wurde der in [8] genauer beschriebene Astronauten-Trainingssimulator gewählt, der in Kooperation des Johnson Space Center in Houston (JSE), der University of Houston Downtown und dem Fraunhofer-Institute for Computer Graphics IGD in Darmstadt entstand.
Üblicherweise wird die Zusammenarbeit zweier Astronauten am JSE direkt vor Ort trainiert. Im Falle einer internationalen Mission ist es allerdings unter Umständen sehr teuer und zeitaufwändig das Training in einer realen Umgebung mit allen Beteiligten durchzuführen. In einem solchen Fall ist es günstiger, das Astronautentraining mit Hilfe einer verteilten virtuellen Umgebung durchzuführen. Also Trainingsszenario wurde die Situation gewählt, an einem Hubble Space Telescop ein defektes Aggregat auszutauschen. Diese Mission war deshalb besonders geeignet, da sie bereits real durchgeführt worden war und aus diesem Grund ausführliches Datenmaterial zur Verfügung stand. Zur Erledigung der Aufgabe werden zwei Astronauten benötigt, von denen einer das defekte Aggregat ausbaut, von seinem Kollegen das Ersatzaggregat entgegennimmt und anschließend wieder einbaut, während der andere das neue Aggregat aus der Ladebucht holt, es übergibt und schließlich das defekte Gerät wieder in der Ladebucht verstaut.
Das Trainingssystem bestand nun aus zwei zeitparallel laufenden VR-Systemen am JSC und dem Fraunhofer-IGD, die beide ihre grafischen Modell-Informationen mittels Inventor-Dateien zur Verfügung gestellt bekamen. Eine ISDN-Standleitung zwischen den Systemen sorgte für die Synchronisation derselben, während die beiden Astronauten über ein analoges Telefon kommunizieren konnten. Als graphisches Ausgabegerät wurde ein Head Mounted Display mit Headtracker verwendet und die Interaktion wurde über einen Datenhandschuh mit integriertem Tracker zur Positionsbestimmung der Hand erfasst.
Zur Steuerung der Figurinen, welche die Astronauten darstellen, wurden autonome Objekte (AO's) eingeführt, die in direktem Zusammenhang mit dem Szenengraphen stehen, wie folgende Abbildung zeigt:

[1] triggern: eine Aktion auslösen



Eine wichtige Operation während der Simulation war das Greifen bzw. wieder Loslassen eines Aggregats, was mithilfe von Kollisions- und Gestenerkennung realisiert wurde, indem ein Objekt einfach gegriffen wird, wenn eine Kollision zwischen Hand und Objekt vorliegt und gleichzeitig eine Faust-Geste zu erkennen ist. Das Loslassen eines Objekts erfolgt einfach indem eine andere Geste als die Faust erkannt wird. Empfängt das AO die Nachricht, dass ein Objekt gegriffen wurde, reagiert es, indem es das assoziierte grafische Objekt im Szenengraph unter die Hand hängt. Erhält es jedoch die Nachricht, dass das Objekt losgelassen wurde, wird selbiges unter die Wurzel der Szene gehängt. Ist ein Objekt gerade "gegriffen", wirkt sich die hierarchische Struktur der Szenengraph-Architektur aus und das Objekt bewegt sich zusammen mit der zugehörigen Hand.
Zur Synchronisation der beiden räumlich getrennten Szenengraphen wurde nun also zum einen die Aktualisierung der Transformation eines grafischen Objekts, das sich bewegt hat, benötigt und zum anderen die Information, dass ein Objekt im Szenengraph "umgehängt" wurde. Dazu war es nötig, die einzelnen Objekte im Szenengraph zu identifizieren. Um dies zu erreichen, erhielt jedes Objekt im Graphen eine eindeutige Identifikationsnummer, die jedoch nicht auf beiden Seiten identisch war. Aus diesem Grund war es nötig, eine Umsetztabelle einzuführen, die allerdings direkt in einen speziellen Renderer des Fraunhofer-Instituts integriert wurde und somit für das Simulationssystem transparent ablief.
Im Großen und Ganzen verlief das Projekt äußerst vielversprechend und wurde von beiden Astronauten als durchaus positiv beurteilt.

Schlusswort

Wenn man die Entwicklung der Virtual und Augmented Reality in den letzten Jahren betrachtet, kann man feststellen, dass dieser Bereich zunehmend vielseitiger und komplexer wird. Man kann allerdings ebenfalls beobachten, dass die Ansprüche an die 3D-Anwendungen steigen. Wärend die Virtuelle Realität vor einigen Jahren noch hauptsächlich für die Spiele-Industrie interessant war, wächst zunehmend auch das Interesse für Anwendungen aus der Medizin, der Archiktektur und vielen weiteren Bereichen. Aus diesem Grund müssen moderne VR- und AR-Umgebungen ein möglichst großes Maß an Präzision und Flexibilität bieten, weshalb wohl auch in Zukunft die Szenengraph-Architekturen eine wichtige Rolle im Zusammenhang mit Virtual und Augmented Reality-Anwendungen spielen werden.

Quellenverzeichnis

Szenengraphen:

[1] Szenengraph-Modellierung
http://ifgivor.uni-muenster.de/vorlesungen/3D_geovisualisierung/NeueMedien2_16.htm

[2] Definition Szenengraph
http://www.gris.uni-tuebingen.de/projects/vis/coursebook/rendering/engine/

[3] Einführung Szenengraphen
http://medien.informatik.uni-ulm.de/lehre/courses/ss02/Computergrafik/DetlefKoentges.pdf

Open Inventor:

[4] Open Inventor
http://www.cg.tuwien.ac.at/studentwork/VRSem96/Inventor/

[5] Einführung in Open Inventor
http://www.ifmi.org/old/full/education/ss2002/vmp/EinfuehrungsBlatt.pdf

[6] Vortrag Open Inventor
http://www.vis.uni-stuttgart.de/img/sommer/seminar/Vortrag-OpenInventor.pdf

[7] Open Inventor
http://www.cg.fb12.uni-siegen.de/Lehre/CG2/Vorlesung%20Material/cg2_12.pdf

Anwendungsbeispiel:

[8] Demonstration von Anwendungen
http://elib.tu-darmstadt.de/diss/000262/6-conclusion.pdf