Eine virtuelle Mall in Java3D

Chris Wewerka, Gerhard Meixner

Fachhochschule Augsburg, Fachbereich Informatik
Infomatec Media AG, D-86167 Augsburg
wewerka@ThreeDimensions.de

Abstract Das Java3D-API kann für alle Arten von 3D-Visualisierungen auf der Java-Plattform verwendet werden. Ausgedehnte virtuelle Welten mit zahlreichen 3D-Objekten, Animationen und Geräuschen stellen das Anwendungsgebiet dar, das in diesem Beitrag näher betrachtet werden soll. Der Funktionsumfang und die Einsatzmöglichkeiten von Java3D werden anhand der Implementierung eines virtuellen Kaufhauses, einer Mall, vorgestellt. Es wird ein Komponentensystem beschrieben, das auf Basis des Java3D-APIs entwickelt wurde, um eine möglichst flexible und komfortable Konfiguration und Ausgestaltung der Mall zu ermöglichen. Die dabei zu Tage getretenen Stärken und Schwächen von Java3D werden diskutiert und dessen Merkmale mit anderen Grafiksystemen, vor allem VRML, verglichen.

Einleitung

Das Java3D-API ist eine Klassenbibliothek für Java, die es erlaubt, in Applikationen oder Applets dreidimensionale Graphik zu verwenden. Es ermöglicht dem Entwickler auf einfache Art und Weise dreidimensionale Körper zu erstellen, zu manipulieren und darzustellen. Die Komplexität eines unter Verwendung von Java3D geschriebenen Programmes kann dabei bis zur Darstellung komplexer virtueller Welten reichen, in denen sich der Besucher frei bewegen und mit seiner Umgebung interagieren kann.
Java3D benutzt und vereinigt sowohl die bewährten Konzepte bestehender Low-Level-Grafiksysteme, wie zum Beispiel OpenGL und Direct3D, als auch von Grafiksystemen auf höherer Abstraktionsebene wie VRML oder OpenInventor. Insbesondere wurde das für die Anwendungsentwicklung in Java3D zentrale Konstrukt Szenengraph von OpenInventor und VRML aufgegriffen. Andererseits überläßt Java3D die Darstellung der 3D-Szenen einer darunter liegenden Low-Level-Grafik-API, nämlich wahlweise OpenGL oder Direct3D. Auch das Beleuchtungsmodell (Lichtquellen, Materialien usw.) lehnt sich sehr stark an OpenGL an.
Java3D bietet mehrere Ausführungsmodi an um 3D-Szenen darzustellen. Die Unterschiede dieser Modi liegen im Grad der Flexibilität bei der Darstellung und Manipulation der 3D-Szene, sowie in der Ausführungsgeschwindigkeit und dem Abstraktionsgrad bei der Gestaltung der Szene.
Der Szenengraph bietet hierbei den größten Komfort für den Entwickler. Ein Szenengraph ist die hierarchische Repräsentation von Objekten, Behaviors (Verhaltensknoten, zuständig für Animation und Interaktion), Licht- und Tonquellen der zu erschaffenden virtuellen 3D-Welt. Er besteht aus Java3D-Objekten (Knoten), die zu einer Baumstruktur zusammengefügt werden. Der Entwickler erzeugt einen oder mehrere Teilgraphen (DAGs) und fügt sie einem virtuellen Universum hinzu. Durch die Anordnung und Verknüpfung der Knoten wird bestimmt, wo und wie die Objekte dargestellt werden und wie der Benutzer mit ihnen interagieren kann.
Java3D verfolgt außerdem den Ansatz, spezielle 3D-Peripherie wie 3D-Mäuse, Head-Mounted-Displays (HMD) oder Shutterbrillen möglichst einfach in das System zu integrieren, indem ihre Funktionalität durch spezielle Schnittstellen abstrahiert wird. Lediglich das Java3D-System ist der Hardware entsprechend zu konfigurieren, bei der eigentlichen Applikation sind jedoch keine Anpassungen notwendig.
Das Ziel der hier vorgestellten Arbeit war es, den Funktionsumfang und die Einsetzbarkeit des Java3D-APIs in einem praxisnahen Umfeld auszuloten. Anhand der Realisierung eines Prototyps für ein virtuelles Kaufhaus entstand ein umfangreicher Erfahrungsbericht für Java3D, auf dessen wesentliche Ergebnisse hier eingegangen werden soll. Im erwähnten Prototyp wurde unter Verwendung der Java3D-Version 1.1.3 eine virtuelle Welt realisiert, welche vom Eintritt des Besuchers bis zum Betreten eines virtuellen Shops reicht, der im Rahmen eines weiteren Projektes [6] entstanden ist.

Ein virtuelles Kaufhaus im Cyberspace : Die 3D-Mall

Im folgenden wird auf die genauen Anforderungen und die zentralen Elemente einer virtuellen Mall eingegangen, die zuerst einmal unabhängig vom verwendeten Grafiksystem definiert werden sollen.

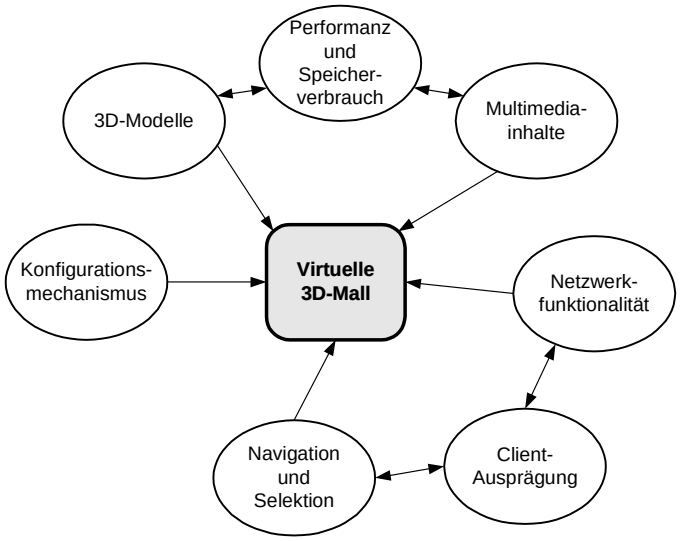


Abb. 1. Aspekte einer virtuellen Mall

3D-Modelle

In einem virtuellen Kaufhaus lassen sich die verwendeten 3D-Modelle zur Ausgestaltung der virtuellen Welt in drei Kategorien einteilen: Modelle zur Erschaffung der virtuellen Welt wie z.B. Gebäude, Modelle, die als interaktive Objekte zur Kommunikation mit dem Benutzer dienen (z.B. Bedienungselemente an Infoterminals oder Aufzugskonsolen), sowie Modelle für die angebotenen Produkte eines virtuellen Kaufhauses. Abhängig vom Aufbau der Mall und den daraus resultierenden Anforderungen an ihre dynamische Konstruktion und das erwünschte Maß an Interaktivität muß für das verwendete Grafiksystem überlegt werden, welche Modelle aus anderen Formaten importiert bzw. konvertiert werden sollen. Mit bestehenden 3D-Modellierungstools wie beispielsweise 3D-Studio-MAX oder Softimage lassen sich auf erheblich einfacherem Weg ansprechende 3D-Modelle erzeugen als in den meisten bestehenden Echtzeit-Grafiksystemen. Somit können die Designer einer virtuellen Welt ihnen vertraute Modellierungsprogramme einsetzen, um die 3D-Welt zu erzeugen, anstatt sie auf wesentlich umständlicherem Weg direkt im Grafiksystem zu programmieren. Bei dem Import bzw. der Konvertierung aus fremden Formaten muß aber beachtet werden, daß dabei oft die Interaktivität bzw. die Dynamik verloren geht. Beispielsweise läßt sich eine automatische Vergrößerung des Mall-Gebäudes aufgrund einer gewachsenen Zahl an integrierten Shops kaum mit einer importierten Lösung realisieren. Bei importierten Objekten, die zur Benutzerinteraktion dienen sollen, muß untersucht werden, ob die gewünschten Methoden (wie z.B. die Selektion) verfügbar sind.

Multimediainhalte

Durch den gezielten Einsatz von Ton und Videoinhalten kann eine virtuelle Mall zu einem audio-visuellen Ereignis für den Besucher werden, wenn beispielsweise das Öffnen einer Türe durch ein passendes Geräusch untermalt oder ein Produkt durch kurze Videoeinspielungen vorgestellt wird. Hierbei ist zu prüfen, inwieweit das zu verwendende Grafiksystem diese Möglichkeiten bietet und wie performant diese sind.

Performanz und Speicherverbrauch

Die Ausführungsgeschwindigkeit ist einer der zentralen Gesichtspunkte bei der Entwicklung einer jeden 3D-Welt, da die Echtzeit-3D-Darstellung hohe Anforderungen an die Hardware des Besuchers stellt. Das ansprechendste Design kann nicht zur Geltung kommen, wenn die Applikation nicht flüssig dargestellt wird. Ziel einer virtuellen Mall im realen Einsatz muß immer die Erreichung eines möglichst großen Zielpublikums sein, und somit muß bei der Entwicklung auf eine performante Darstellung geachtet werden. Das bedeutet, daß ein großes Augenmerk auf Optimierungsmöglichkeiten innerhalb der Grafiksysteme gelegt werden muß.
Der Speicherverbrauch einer 3D-Anwendung läßt sich in zwei Bereiche einteilen: Erstens den Verbrauch an Netzbandbreite, die bei der Übertragung sämtlicher Daten wie z.B. Texturen, Multimedia-Inhalte und Geometrieinformationen benötigt wird und zweitens den Verbrauch an Hauptspeicher, der zum Ablauf auf dem Client-Rechner gebraucht wird. Während die Daten meist in einem komprimierten Format übertragen werden

können, müssen sie auf dem Client-Rechner unkomprimiert oder sogar in einem ablaufoptimierten, noch speicherintensiveren Format vorliegen. Hier existieren bei den gängigen Grafiksystemen Trade-Offs zwischen Speicherverbrauch und Performanz, wie zum Beispiel bei dem LOD-Prinzip (Level Of Detail), oder dem sogenannten Mip-Mapping bei Texturen. Die vorgegebene Netzwerkbandbreite erzeugt in der Regel die stärksten Restriktionen für die Entwicklung einer virtuellen Mall, da dem Besucher mit einer Netzwerk-Standardhardware nur eine begrenzte Wartezeit bei der Datenübertragung zugemutet werden kann.

Netzwerkfunktionalität

Eine virtuelle Mall ist eine klassische Multiuser-Anwendung für das Internet: Ihre Besucher können sich durch die virtuelle Welt bewegen, dabei auch andere Besucher sehen und sich mit ihnen austauschen. Weiterhin sind auch virtuelle Berater denkbar, die dem Besucher die Mall näher erklären. In diesem Zusammenhang spielt das dem Grafiksystem zugrunde liegende Progammierkonzept eine entscheidende Rolle. Welche Mechanismen zur Kommunikation über das Internet werden geboten, wie performant und wie simpel bzw. wie abstrahiert sind diese in der Anwendung und wie gut kooperieren das 3D-Grafiksystem und die Programmierschnittstelle?

Navigation und Selektion

Die Navigation und die Selektion von 3D-Objekten stellen die interaktiven Komponenten einer virtuellen Mall dar. Die Navigation in einer virtuellen Welt verdient besondere Beachtung. Nur durch eine intuitive und unkomplizierte Art der Bewegungssteuerung kann der Besucher die virtuelle Welt wirklich erleben. Hierbei können und sollten auch Mechanismen gewählt werden, die dem Benutzer die Navigation durch eine geführte Bewegung erleichtern. Ein unzureichendes oder wenig intuitives Navigationssystem kann sehr schnell dazu führen, daß der Besucher die virtuelle Welt frustriert verläßt, beispielsweise weil er eine bestimmte Tür nur nach mehrmaligen Versuchen passieren kann.
Die Auswahl von 3D-Objekten dient der Interaktion des Besuchers mit Gegenständen der virtuellen Welt. Durch die Realisierung der Selektion können beispielsweise Produkte in 3D-Darstellung ausgewählt und von allen Seiten betrachtet werden. Die verschiedenen verfügbaren 3D-Grafiksysteme unterscheiden sich in diesem Bereich sehr im Hinblick auf Flexibilität und Abstraktionsgrad.

Client-Ausprägung

Durch die Clients müssen alle mit der Mall zusammenhängenden Daten und gegebenenfalls auch Programmcode vom Server heruntergeladen, die Szenen gerendert und für die Interaktion eine Verbindung zum Server unterhalten werden. Diese Aufgaben können mit einer eigenen Client-Installation, Applets oder mit einem speziellen Browser-Plugin (wie bei VRML) erledigt werden. Letztere bringen zum Teil bereits Netzwerkfunktionalität, ein Navigationssystem sowie einen Selektionsmechanismus mit, was jedoch wiederum die Flexibilität in der Anwendungsentwicklung einschränken kann.

Konfigurationsmechanismus

Ein flexibler und möglichst einfacher Konfigurationsmechanismus ist das Kernstück einer virtuellen Mall. Administratoren oder Designer der Mall sollten nur geringe Vorkenntnisse im Bereich der Computergrafik besitzen müssen, um die Mall entsprechend ihrem Aufgabengebiet zu konfigurieren bzw. auszugestalten. Der Konfigurationsmechanismus muß die Möglichkeit bieten, Multimediainhalte und 3D-Modelle (z.B. für Produkte) in die Mall zu integrieren. Weiterhin werden die zur Verfügung stehenden Shops durch eine entsprechende Konfiguration in die Mall integriert, wodurch ein dynamischer Aufbau der Mall erreicht wird. Dazu bedarf es natürlich einer erweiterbaren Grundstruktur der Mall.

Umsetzung einer Mall in Java3D

In diesem Kapitel werden die Möglichkeiten, die das Java3D-API in der Version 1.1.3 bietet anhand einiger zentraler Punkte der Implementierung der 3D-Mall beschrieben. Bis auf die Netzwerkfunktionalität wurden die wesentlichen Aspekte von Abbildung 1 umgesetzt. Die bei der konkreten Benutzung des APIs gewonnenen Erfahrungen und Erkenntnisse sollen nun herausgestellt werden.

Die Mall mit den eingebundenen Shops ist futuristisch, im Science-Fiction-Stil gestaltet. Sie dient als zentraler Einstiegspunkt für den Besucher. Er erhält hier einen Überblick über die in die Mall integrierten Shops und es werden ihm bereits ausgewählte Produkte aus einzelnen Shops präsentiert. Die frei wählbare Zahl der in die Mall einzufügenden Shops beeinflußt die Form der Mall. Mehrere Shops können in der Mall zu Kategorien (z.B. Sportgeschäfte) zusammengefaßt werden. Diese werden, wie die Shops selbst, geometrisch durch Hedraeder repräsentiert. Von den Kategorie-Knoten kann der Besucher in verbindenden Gängen zu weiteren Hedraedern gelangen, die untergeordnete Kategorien repräsentieren oder schließlich Shops beherbergen. Der Innenraum eines Hedraeders ist durch drei Plattformen unterteilt, auf denen sich der Besucher bewegen kann. Mittels eines Aufzuges kann er die gewünschte Plattform erreichen. Der Aufbau der Mall (Anzahl und Art der Shops und deren Zusammenschluß zu Kategorien) und die Abmessungen der Hedraeder für die Kategorien und die Shops, sowie die gesamte Ausstattung der Mall (3D-Modelle, Produkte, Geräusche, Lichtquellen, Animationen u.v.m.) sind flexibel konfigurierbar. In der aktuellen Implementierung erfolgt diese Konfiguration über Property-Dateien, jedoch ist eine Umstellung auf eine Datenbank programmtechnisch vorgesehen.

Das Herzstück der flexiblen Konfiguration : Das Komponentensystem der 3D-Mall

Komponenten in der 3D-Mall sind flexibel einsetzbare und konfigurierbare Elemente, aus denen der Szenengraph aufgebaut werden kann. Beispielsweise ist ein aus VRML nach Java3D importiertes Objekt eine Komponente. Rotationskomponenten, die wiederum andere Komponenten animiert rotieren lassen können, liefern ein weiteres einfaches Beispiel. Komponenten stellen damit eine Abstraktionsschicht dar, die auf dem Szenengraph von Java3D aufsetzt und durch dessen klaren hierarchischen Aufbau erst möglich wird. Durch Komponenten wird somit die Mall mit Objekten und Multimediainhalten ausgestattet. Besonders weitreichend ist die Möglichkeit, Komponenten für interaktive Bestandteile der Mall zu verwenden. Beispielsweise wurde ein Aufzug zwischen den Plattformen der Mall realisiert, welcher als funktionalen Kern eine entsprechende Schaltkonsole besitzt. Die visuelle Repräsentation von Aufzug und Konsole bleibt frei konfigurierbar.
Komponenten werden durch den Konfigurationsmechanismus, z.Zt. über Property-Dateien, in die Mall integriert und wie gewünscht konfiguriert. Komponenten können dabei auch ineinander geschachtelt werden. So kann eine Komponente, die ein 3D-Objekt kapselt, durch eine übergeordnete Rotationskomponente animiert rotiert werden, die wiederum durch eine Transformationskomponente an die gewünschte Position verschoben wird. Dabei bleibt der Zugriff auf die zur Interaktion verwendeten 3D-Objekte trotz Schachtelung weiterhin über eine Schnittstelle der Komponenten möglich.
Eine Komponente kann an vielen Stellen der Mall eingesetzt werden. Die Zuordnung ihrer Funktion und ihrer Position (z.B. als Türschild, Aufzug bzw. Aufzugskonsole, Infoterminal, Objekt auf einer Plattform) erfolgt durch entsprechende Verweise auf die Komponente in den Konfigurationsdateien der Mall, in denen die Komponente auch definiert und konfiguriert wird. Die Mall bietet einen vordefinierten Satz an Komponenten. Sie kann darüber hinaus durch benutzerdefinierte Komponenten erweitert werden, die dynamisch nachgeladen werden.
Im folgenden soll die Einbindung einer einfachen, geschachtelten und interaktiven Komponente, ein rotierendes Bedienungselement mit zwei Knöpfen, in den Szenengraph der Mall demonstriert werden. In Abbildung 2 werden rechts die erforderlichen Einträge in den Property-Dateien angegeben, die den links dargestellten Teil des Szenengraphen erzeugen. Die VRML-Komponente ist die innerste Komponente und beinhaltet neben reinen Darstellungsobjekten (Shape), die beiden zur Selektion verwendeten 3D-Objekte `Button1` und `Button2`. Die Rotator-Komponente erzeugt mit ihrem `RotationInterpolator` die Rotations-Animation durch kontinuierliche Modifikation der 3D-Transformation der `TransformGroup`, die der VRML-Komponente übergeordnet ist. Die übergeordnete Konsolenkomponente überwacht schließlich durch die Verwendung der `PickBehavior` (siehe hierzu auch nächster Abschnitt) ihre enthaltene Rotator-Komponente auf eventuell auftretende Selektionsereignisse der 3D-Objekte `Button1` und `Button2`, bei deren Eintreten sie entsprechende Aktionen auslöst. Dieses einfache Beispiel für die Erstellung und Konfiguration einer geschachtelten Komponente demonstriert die Flexibilität des Komponentensystems.

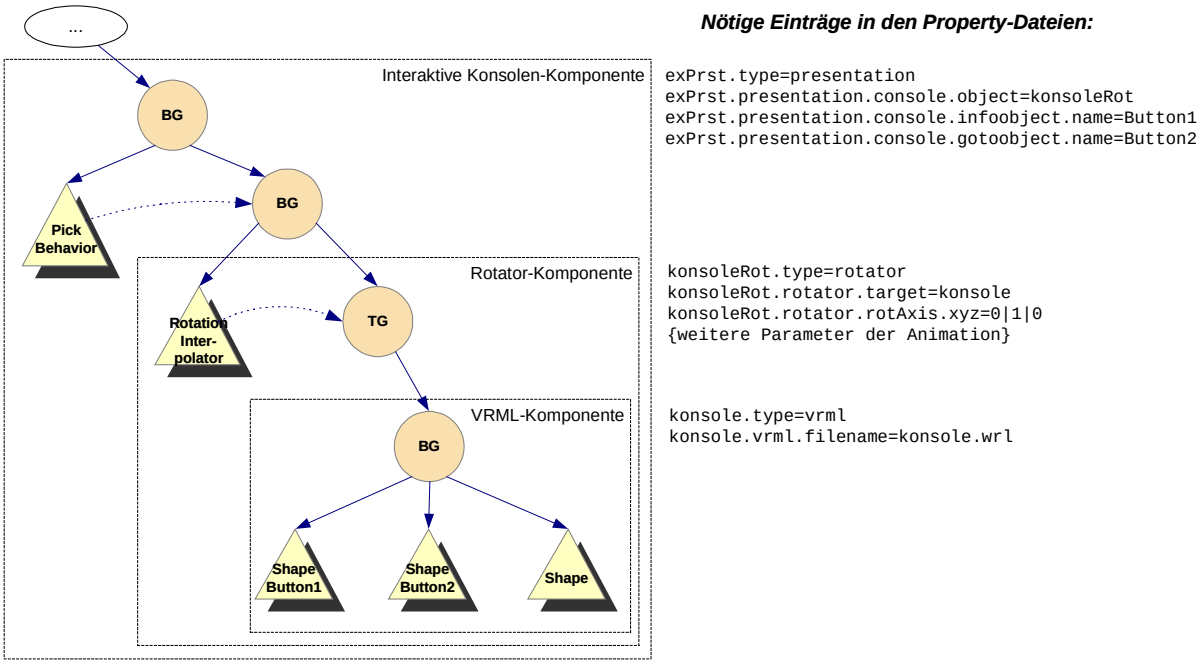


Abb. 2. Integration einer Komponente in die Mall

Nachfolgend werden zwei komplexere Einsatzmöglichkeiten und Konfigurationsbeispiele für Komponenten kurz beschrieben:

Die Komponente Aufzug

Der Aufzug der Mall, der dazu dient, den Besucher auf die verschiedenen Plattformen zu transportieren, zeigt das Zusammenspiel des Komponentenkonzepts mit einigen wichtigen Elementen von Java3D. Nähert sich der Besucher dem Aufzugsobjekt, wird das Bedienpult des Aufzuges eingeblendet, wodurch der Besucher seinen aktuellen Standpunkt angezeigt bekommt und auch die Möglichkeit hat, durch Auswählen auf eine neue Plattform zu gelangen. Weiterhin wird die Funktion des Aufzuges und der Konsole durch Animationen verdeutlicht.
Die Animation, die Annäherungserkennung, sowie die Selektion werden in den entsprechenden Komponenten durch das Java3D-Konzept der Behaviors realisiert. Behaviors sind Elemente des Szenengraphen, die zur Interaktion und Animation eingesetzt werden. Sie werden bei Eintreffen spezieller Ereignisse aktiv, um dann eventuell Änderungen am Szenengraphen durchzuführen. Stärken dieses Konzeptes sind die mögliche Kapselung des Animations- bzw. Interaktionscodes, sowie die Einbettung in den Szenengraphen.

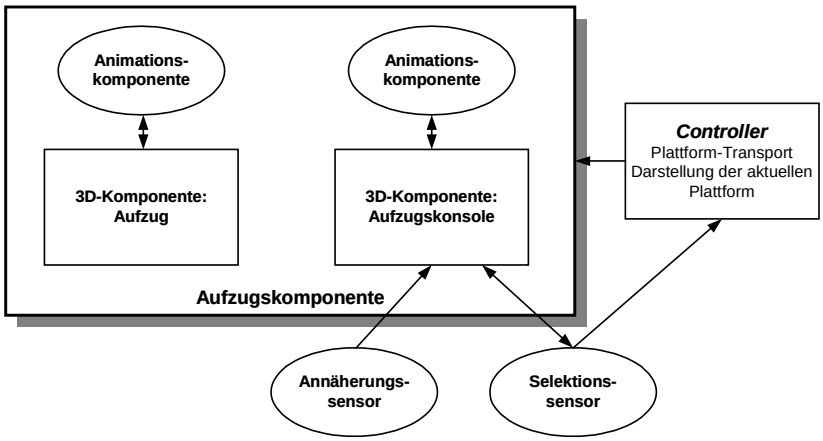


Abb. 3. Struktur der interaktiven Aufzugskomponente

Der Aufzug verwendet also mehrere einfache Java3D-Behaviors, um eine komplexere Anwendungssemantik zu realisieren. Die Annäherungs-Behavior des Aufzugsobjektes steuert die Einblendung der Konsole, während die Selektions-Behavior die Auswahl der gewünschten Plattform steuert. Animations-Behaviors werden schließlich verwendet, um die Aufzugsobjekte zu animieren und den Besucher zur gewünschten Plattform zu transportieren.

Die visuelle Repräsentation, ebenso wie die Animation des Aufzuges und seiner Konsole ist konfigurierbar und wird in der Demoversion durch importierte VRML-Objekte realisiert. Auf dem teilweise transparenten Aufzug sind animierte Schlieren zu erkennen, die durch eine benutzerdefinierte Behavior realisiert werden, die fortwährend die Texturtransformation des 3D-Objektes modifiziert. Die notwendige Zuordnung der „Knöpfe" der Konsole zur entsprechenden Selektions-Behavior erfolgt per namentlicher Referenzierung durch den Konfigurationsmechanismus. Weiterhin speichert eine zentrale Controller-Klasse die aktuelle Plattform des Besuchers und initiiert den Transport des Besuchers zwischen den Plattformen.

Die Komponente PresentationContainer

Instanzen von `PresentationContainer` dienen der Präsentation von Werbung, wie zum Beispiel Werbetafeln oder Sonderangebote einzelner Shops in der Mall. Der Besucher soll damit auf einzelne Produkte schon in der Mall hingewiesen und zum Besuch des zugehörigen Shops angeregt werden.

Der `PresentationContainer` selbst enthält wiederum zwei Komponenten: eine Komponente, die das Werbeobjekt und eine weitere, die das zugehörige Bedienelement darstellt. Durch letzteres kann der Besucher textuelle Informationen über das Produkt in Form eines Tooltips abrufen oder sich direkt zum anbietenden Shop transportieren lassen. Diese 3D-Objekte können im Rahmen der Mall-Konfigurierung beliebig animiert werden. Zu diesem Zweck stellt die Mall einige vordefinierte Animationskomponenten zur Verfügung, welche zum Teil mit vordefinierten Java3D-Behaviors, zum Teil aber auch mit selbst entwickelten Behaviors realisiert wurden. Beispiele sind die animierte Rotation, das animierte „Morphen" von mehreren Objekten, sowie die schon erwähnte Animation der Texturtransformation.

Durch den Einsatz von Soundkomponenten, die sich des Java3D-Raumklang-Soundsystems, d.h. der dazu vorhandenen Szenengraphknoten bedienen, läßt sich die Shopwerbung durch Musik und sonstige Samples unterlegen. Deren Raumklang-Attribute (z.B. Hall, Dämpfung) lassen sich sehr komfortabel in Java3D konfigurieren. Leider können momentan noch keine Filme in die 3D-Welt integriert werden, da die entsprechende Funktionalität noch nicht in Java3D implementiert ist.

VRML-Import

Aufgrund des komponentenorientierten Aufbaus der Mall bietet es sich an, geeignete 3D-Objekte als Komponenten aus 3D-Modellier-Programmen wie beispielsweise 3D-Studio Max zu importieren. Mit dieser einfachen und schnellen Möglichkeit können Komponenten auch ohne Programmierkenntnisse erstellt werden. Java3D unterstützt durch ein Mini-API den Import von Objekten oder ganzen 3D-Szenen aus anderen Formaten. Externe Entwickler-Gruppen haben bereits eine Vielzahl von Import-Modulen für die gängigen Formate entwickelt, wobei zum gegenwärtigen Zeitpunkt der VRML-Loader des Web3D-Konsortiums [7] als mit Abstand am ausgereiftesten gelten kann. Mit Ausnahme von Animationen können hier die Szenen so importiert werden, daß keine bzw. nur geringfügige Modifikationen erforderlich sind.

Der Lader erzeugt aus der VRML-Welt einen entsprechenden Java3D-Teilgraphen, der nach erfolgtem Import noch analysiert und modifiziert werden kann. Die für den VRML-Import zuständige Mall-Komponente ermöglicht auf diesem Weg unter anderem die Verwendung des Java3D-Environment-Mappings für importierte VRML-Objekte, ein Feature, das in VRML selbst nicht unterstützt wird. Importierte Objekte können, wie schon erwähnt, auch zur Benutzerinteraktion verwendet werden. In diesem Fall muß die Komponente ebenfalls nachbearbeitet werden, um sie in den Picking-Mechanismus mit einzubeziehen.

Integration von APIs

Dem Java3D-Entwickler wird durch das breite Spektrum an verfügbaren Erweiterungs-APIs für die Java-Plattform eine Vielzahl von zusätzlichen Erweiterungsmöglichkeiten für seine Applikation geboten. Bei der Erstellung des 3D-Mall-Prototyps wurde die Java Advanced Imaging API (kurz JAI) verwendet, um eine Unterstützung für PNG-Bilder zu integrieren. Der Vorteil des PNG-Formates liegt in einem 256-Bit Alpha-Kanal, der zur Erzeugung feiner Transparenzübergänge verwendet werden kann. Durch die JAI könnten weiterhin Texturen für 3D-Objekte dynamisch erstellt, bearbeitet oder durch gewünschte Filter modifiziert werden. Mittels der Swing-API wurde ein HTML-Hilfesystem integriert, das aus der 3D-Mall heraus aufgerufen werden kann. Weitere naheliegende, noch nicht realisierte Anbindungen sind z.B. die Integration der JavaMail-API zum Mailversand/-empfang innerhalb der Mall und vor allem die Integration von Videos über das Java Media Framework (JMF).

Die Navigation und die Benutzerführung

Ein intuitives und möglichst einfach zu bedienendes Navigationssystem stellt einen der wichtigsten Akzeptanz-Aspekte von virtuellen Welten dar. Zu eng dimensionierte oder mit Objekten überfrachtete Welten sollten in diesem Zusammenhang vermieden werden, da sie die Navigation unnötig erschweren.
Java3D bietet von Haus aus kein vordefiniertes Navigationssystem durch ein bestimmtes Eingabegerät, was in puncto Flexibilität ein großer Vorteil ist. Während in VRML dem Browser, die nicht immer gelungene Navigationsschnittstelle überlassen ist und somit keine Modifizierungsmöglichkeit für den Entwickler der virtuellen Welt besteht, kann die Navigation durch ein beliebiges Eingabegerät in Java3D vollständig selbst implementiert werden. Daß dazu kein extrem hoher Aufwand nötig ist, zeigt die knappe, aber effektive und wiederverwendbare Realisierung der Navigation in der Mall mittels der Maus (ca. 200 Zeilen Java-Sourcecode). Die Realisierung erfolgt hierbei über die oben schon erwähnten Java3D-Behaviors, die die entsprechenden Mausereignisse auf Bewegungen des virtuellen Standpunktes des Besuchers umrechnen. Besondere Positions-Eingabegeräte wie z.B. Headtracker o.ä. lassen sich durch das Java3D-View-Modell einfach in ein System integrieren. Der zentrale, neuartige Gedanke ist hierbei die strikte Trennung der Behandlung der Eingabegeräte vom eigentlichen Applikationscode. Eingabe- und auch Ausgabegeräte wie z.B. Multi-Monitor-Systeme, Caves oder 3D-Stereo-Brillen sind in Java3D Konfigurationsparameter der Systemumgebung und nicht der eigentlichen Applikation.
Die gesamte Navigation in der Mall besteht grundsätzlich aus drei verschiedenen Fortbewegungsmöglichkeiten: Die schon erwähnte Steuerung mit der Maus, als zweites ein sogenannter automatischer Transport, der bei Annäherung an bestimmte Punkte der 3D-Welt (z.B. Türen zu Verbindungsgängen) automatisch ausgelöst wird und den Besucher entlang einer berechneten Bahn führt. Diese Art der Fortbewegung ist für Bereiche geeignet, die zwar aus konstruktions- und designtechnischen Gründen in die Welt integriert werden sollen, jedoch keinen Mehrwert bringen, wenn der Besucher sie manuell erforschen kann, wie zum Beispiel Gänge zwischen Gebäuden. Die dritte Möglichkeit der Fortbewegung dient dem schnellen Erreichen eines bestimmten Punktes und wird durch das Applikationsmenü und einige spezielle Bedienelemente wie z.B. das Bedienelement eines `PresentationContainers` durchgeführt, wobei der Besucher direkt, d.h. ohne Animation seiner Position an den gewünschten Zielpunkt transportiert wird.
Alle Navigationsmöglichkeiten der Mall modifizieren die 3D-Transformation, die das zentrale Element des View-Modells, die View-Plattform, in der virtuellen Welt positioniert. Diese Plattform stellt man sich am besten als fliegenden Teppich vor, auf dem der Besucher steht, sich aber auf ihm beliebig bewegen oder seinen Kopf drehen kann. Um den Benutzer durch die Welt zu führen, „fliegt“ die Anwendung den Teppich an die gewünschte Position. Die Berechnung der Zwischenschritte für die geführte Bewegung des Besuchers, innerhalb der entsprechenden Animations-Behavior, wird in der Mall mittels linearer Interpolation zwischen Start-, Zwischen- und Endpositionen und einer sogenannten Quaternion-Interpolation (SLERP) für die Berechnung der Zwischenschritte der Orientierung, d.h. quasi der Blickrichtung des Besuchers durchgeführt. Vorteil des View-Plattform-Ansatzes gegenüber konventionellen kamerabasierten Ansätzen ist eine einfache Integration von 3D-Eingabegeräten wie z.B. Headtrackern, da die gewünschte Blickrichtung und Position des Besuchers in einer von der anwendungsseitig verwendeten Transformation für die Positionierung und Orientierung der Plattform getrennt wird.

Kollisionssystem
In einer virtuellen Welt ist das Erkennen von Kollisionen des Besuchers mit „im Wege stehenden“ 3D-Objekten ein wichtiger Bestandteil der Navigation. Das für diesen Zweck in Java3D implementierte Konzept leidet zur Zeit noch an einigen Fehlern und Einschränkungen, auf die kurz im letzten Kapitel eingegangen wird. Dies machte es notwendig, für die Umsetzung der Mall eine alternative, performante Lösung zu entwickeln. Die gewählte Methode könnte man am besten als Kollisionsvermeidung bezeichnen: Das Navigationssystem, das die Bewegung des Betrachters mittels einer oder mehrerer Eingabegeräte realisiert, besitzt eine Datenstruktur (eine Art Karte), die spezifiziert, in welchen Bereichen der Besucher der Welt sich bewegen darf (alternativ dazu die Angabe der verbotenen Bereiche). Führt er eine Bewegung mittels seines Eingabegerätes durch, wird zuerst überprüft, ob sich der Besucher weiterhin in dem zulässigen Bereich befindet und falls dies zutrifft, wird die Bewegung wirklich durchgeführt. Würde er einen unzulässigen Bereich betreten, würde die Bewegung hingegen nicht ausgeführt. Als Erweiterung könnte hier ein Gleiten entlang des unzulässigen Bereiches implementiert werden. Um die Performanz bei der notwendigen Bereichsprüfung zu steigern, wird die virtuelle Welt in mehrere Gebiete unterteilt, in denen jeweils gleichartige zulässige bzw. unzulässige Bereiche gelten. Somit werden in den entsprechenden Navigations-Behaviors nur die relevanten Bereiche für das Gebiet, in denen sich der Besucher gerade aufhält, überprüft. Eine zentrale Instanz erkennt den Wechsel der Gebiete und übergibt dem Navigationssystem die neu geltenden Bereiche. Soll der Betrachter auf dem Boden laufen, kann mittels Aussendung eines Strahles nach unten und nachfolgender Ermittlung des Auftreffpunktes bei jeder Bewegung die Höhe des Betrachters in der virtuellen Szene entsprechend angepaßt werden.

Diese Lösung hat sich in puncto Ausführungsgeschwindigkeit als sehr effizient erwiesen, jedoch ist gegenüber einer automatischen Erkennung von Kollisionen aufgrund der Geometrie der Objekte ein Mehraufwand für die Erstellung der Datenstruktur für die Kollisionskarte notwendig.

Optimierung der Ausführungsgeschwindigkeit und des Speicherverbrauchs

Wie in anderen 3D-Grafiksystemen auch hat die Komplexität der darzustellenden Szene den größten Einfluß auf die Ausführungsgeschwindigkeit der Java3D-Applikation. D.h. zum Beispiel die Anzahl der Licht- und Tonquellen und vor allem die Polygonzahl aller darzustellenden Objekte beeinflussen die Laufzeiteigenschaften des 3D-Programmes enorm. Folgendes Diagramm, das durch Messungen an einer animierten Java3D-Szene (rotierendes Objekt durch zwei Punktlichtquellen, einer gerichteten und einer ambienten Lichtquelle beleuchtet) entstand, verdeutlicht den engen Zusammenhang zwischen Polygonzahl und der Ausführungsgeschwindigkeit:

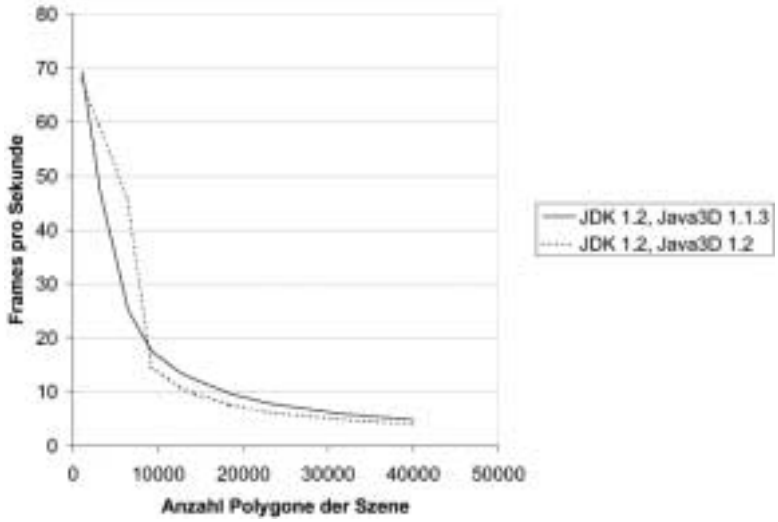


Abb. 4. Performanz von Java3D in Abhängigkeit von der Polygonzahl

Der Benchmark wurde auf einem PII-400 mit 128MB RAM unter Windows NT SP6 mit einer Riva TNT 3D-Grafikkarte durchgeführt. Dabei ist anzumerken, daß durch die Verwendung des JDKs 1.3 und deren integrierter HotSpot-Technologie kein Geschwindigkeitszuwachs erzielt werden konnte. Es wird deutlich, daß die Framerate ab ca. 18000 Polygonen auf für den Benutzer inakzeptable 10 fps fällt. Die Anzahl und die Art der Lichtquellen beeinflussen die Ausführungsgeschwindigkeit ähnlich stark. Eine Punktlichtquelle reduziert die Framerate bei dem verwendeten Testprogramm um ca. 10-20% gegenüber der Verwendung einer gerichteten Lichtquelle. Im folgenden wird auf einige vorhandene Optimierungsmöglichkeiten des Java3D-APIs eingegangen, die auch in der Mall eingesetzt wurden.
Durch den Einsatz von Level-Of-Detail-Techniken (möglich durch die Behavior `LOD`), die es ermöglichen je nach Entfernung zu einem bestimmten Punkt unterschiedliche Teilbäume des Szenengraphen darzustellen, lassen sich Objekte je nach ihrer Distanz zum Betrachter in einer unterschiedlichen Genauigkeit darstellen oder vollständig aus dem Szenengraphen entfernen. Die Entfernung von vollständig verdeckten oder nicht sichtbaren Objekten (beispielsweise die Einrichtung eines Raumes, der von außen betrachtet wird) hat sich bei der Entwicklung der Mall als sehr sinnvoll erwiesen, da selbst diese einen hohen Anteil an Rechenzeit verbrauchen, weil fast alle nötigen Berechnungen der Rendering-Pipeline auch für diese Objekte durchlaufen werden. Erst bei ihrer Darstellung unter Verwendung des Z-Buffers (Tiefenbuffer) werden die Bildpunkte dieser Objekte durch Punkte von näher zum Betrachter gelegenen Objekten überschrieben.
Mittels der sogenannten Capability-Bits der Szenengraph-Elemente kann der 3D-Anwendungs-Entwickler dem Java3D-System mitteilen, an welchen Stellen Optimierungen des Graphen vorgenommen werden können. Werden beispielsweise die 3D-Transformationen mehrerer übereinander angeordneter Transformationsgruppen nur bei ihrer Instanziierung gesetzt und nie wieder verändert, kann das Java3D-System diese zu einer einzigen `TransformGroup` mit einer einzigen Transformation „ausflachen“. Die Performanz der 3D-Mall konnte durch diese Optimierungsmöglichkeit jedoch nur minimal gesteigert werden.
Behaviors sind, wie schon erwähnt, Szenengraphelemente, die hauptsächlich zur Animation und Interaktion verwendet werden. Sie können auf bestimmte Ereignisse reagieren und daraufhin andere Elemente der 3D-Szene modifizieren. Bei einer großen virtuellen Welt mit zahlreichen Animationen, die alle gleichzeitig ablaufen würden, wäre somit der Rechner schnell überfordert. Java3D verwendet deswegen `SchedulingBounds` bei seinem Behavior-Konzept. Diese Bereiche legen Gebiete in der virtuellen Welt fest, die die zugehörige Behavior

erst dann aktivieren (d.h. für den Empfang von Ereignissen aktivieren), wenn der Besucher das Gebiet betritt. Durch sinnvolles Festlegen dieser Bereiche für die Animationen der Mall konnte hier eine extreme Steigerung der Geschwindigkeit der Applikation erzielt werden.
Da 3D-Anwendungen ebenso rechenzeit- wie auch speicherintensiv sind und der Mangel an physikalischem Hauptspeicher durch den Auslagerungsmechanismus des Betriebssystems zur Verlangsamung der Applikation führt, wurden in Java3D zwei Mechanismen implementiert, die das Wiederverwenden von Teilbäumen des Szenengraphen ermöglichen. Der erste Mechanismus kann durch spezielle `SharedGroup`-Knoten verwendet werden und ermöglicht die gemeinsame Verwendung eines gesamten Teilgraphen. Dieser Mechanismus gleicht dem VRML DEF/USE-Konzept und wurde in der Mall aufgrund mangelnder Flexibilität nicht eingesetzt. Der zweite Mechanismus, das Klonen von Teilgraphen, bietet sich an, wenn nur speicherintensive Teile (die sog. `NodeComponents` wie z.B. Geometriedaten und Texturen) des Szenengraphen wiederverwendet werden sollen, die Struktur oder einige andere `NodeComponents` des gemeinsam benutzten Graphen jedoch individuell änderbar sein sollen. Der Entwickler hat hierbei die Möglichkeit genau festzulegen, welche `NodeComponents` mehrfach referenziert oder statt dessen kopiert werden. Durch diesen Mechanismus konnte der Speicherverbrauch der 3D-Mall erheblich reduziert werden, da er auch bei den häufig verwendeten importierten Komponenten verwendet wird.

Die Selektion in der Mall

Die Selektion (auch Picking genannt) von 3D-Objekten in der Mall wird auf der Grundlage von Hilfsklassen des Java3D-Paketes realisiert. Umgesetzt wird das Picking durch Aussenden eines Strahles vom Standpunkt des Betrachters aus in die virtuelle Welt, dessen Richtung durch die Klickposition bestimmt wird. Das erste Objekt, das der Strahl erreicht und dessen Selektionsbit aktiviert ist, wird als Ergebnis des Pickingvorganges zurückgeliefert. Für das Empfangen der Mausereignisse werden ebenso wie bei der Navigation Behaviors verwendet. Diese Vorgehensweise zeichnet sich durch ein hohes Maß an Flexibilität aus: Das Picking in Java3D kann individuell an die gewünschte Situation angepaßt werden. Beispielsweise können anstatt eines Pickingstrahles ohne Ausdehnung andere Objekte zur Selektion verwendet werden, wie z.B. ein Kegel. Ebenso können alle vom Picking-Objekt getroffenen 3D-Objekte der Szene ermittelt werden. Die Anbindung von spezieller 3D-Eingabe-Hardware wie z.B. Datenhandschuh oder 3D-Maus wird dadurch weiterhin relativ einfach möglich (zusätzlich unterstützt durch das Interface `InputDevice`, dessen Implementierungen im Java3D-System registriert und dadurch einfach integriert werden können). Abstraktere Lösungen wie in VRML sind zwar etwas einfacher anzuwenden, für viele Anwendungsgebiete jedoch zu unflexibel.

Konzeptionelle und implementierungsbedingte Merkmale von Java3D

Integration in Java im Vergleich zu VRML

Im WWW-Umfeld tritt Java3D in direkte Konkurrenz zu dem 3D-System VRML. Dabei ist eine der großen Stärken des Java3D-APIs die Integration in die Java-Plattform. Durch Java3D wird es möglich in Java performante (auch die zur Verfügung stehende 3D-Hardware ausnutzende) 3D-Applikationen, wie die 3D-Mall, zu entwickeln, die zudem noch von der Vielfalt an bestehenden APIs für die Java-Plattform profitieren können, welche direkt mit der Java3D-Anwendung verschmelzen können. Hier zeigen sich große Vorteile gegenüber VRML:

- Objektorientierung: Das Klassenkonzept inklusive Vererbung und Kapselung ermöglicht die Wiederverwendung und Erweiterung von Szenengraphelementen in Java3D. VRML-Prototypen ermöglichen es demgegenüber, nur grafische Elemente zu Kollektionen zusammenzufassen, Funktionalität kann nicht vererbt werden.
- Typsicherheit: Java3D besitzt durch Java die Möglichkeit zur typsicheren Kommunikation, in VRML ist nur eine simple Kommunikation mittels `Routes` möglich, es läßt sich jedoch nicht einmal der Sender eines Ereignisses z.B. zum Zweck der bedingten Ausführung prüfen.
- Strukturierung: Da VRML keine Programmiersprache enthält, werden durch das VRML-Scripting oder das External Authoring Interface (EAI) Programmiersprachen wie JavaScript oder Java für die Belange eingesetzt, die nur mit einer solchen Sprache gelöst werden können. Der Einsatz von Scriptsprachen wie z.B. JavaScript ist auch bei kleinen Anwendungen sehr schnell nötig. Komplexe Anforderungen, wie sie zum Beispiel eine Multiuser-Welt darstellt, erfordern das EAI, wodurch ein hybrides System aus grafischer Repräsentation (VRML-Welt) und der entsprechenden Programmlogik (Netzwerkfunktionalität u.v.m. in Java) entsteht, das schlecht wartbar und erweiterbar ist. Zudem werden VRML-Browser und zugehöriges

EAI-Applet, welches eventuell das 2D-Interface darstellt, immer nebeneinander existieren. Möglichkeiten, wie man sie in Java mit Java3D hat, wie z.B. das Darstellen der 3D-Szene als Teil eines Applets, sind generell nicht möglich.

- Animation und Interaktion: Das Behavior-Konzept von Java3D, das vor allem für Animation und Interaktion eingesetzt wird, demonstriert die Mächtigkeit des objektorientierten Ansatzes: Einmal entwickelte Behavior-Funktionalität wird in einer Klasse gekapselt und kann in abgeleiteten Klassen durch Vererbung im gewünschten Maße wiederverwendet werden. Hierbei bleibt dem Entwickler jedoch in vielen wichtigen Einsatzgebieten der Behaviors (z.B. Navigation, Selektion) im Gegensatz zu VRML die nötige Flexibilität erhalten. Eine weitere Stärke der Behaviors ist ihr Registrierungsmechanismus für die relevanten Ereignisse am Java3D-System. Das zentrale Kommunikationselement von VRML, die Routes, sind dazu im Vergleich unstrukturiert und bei umfangreichen 3D-Welten absolut unübersichtlich.
- Performanz: Java3D bietet die Möglichkeit gezielt den Szenengraph zu optimieren, je nachdem, welche Zugriffe während des Programmablaufs auf den Graphen nötig sind (Capabilities), wodurch sich eine höhere Ausführungsgeschwindgkeit erreichen läßt. VRML-Welten werden dagegen interpretiert abgearbeitet und erlauben keine wirkungsvollen Optimierungen, da die Struktur des Szenengraphen unter anderem für externe Zugriffe durch das EAI beibehalten werden muß.

Schwächen der derzeitigen Implementierung

Java3D besitzt in der derzeitigen Implementierung (1.2), abgesehen von der noch fehlenden Unterstützung für Video-Wiedergabe in der 3D-Welt (ähnlich den `MovieTextures` in VRML) und der fehlenden Export- bzw. Speicherungsmöglichkeiten, zwei große Schwächen: Die integrierte Kollisionserkennung, sowie die fehlerhafte Darstellung mehrerer hintereinander liegender transparenter Objekte.

Kollisionserkennung

Unter Kollisionserkennung ist die Erkennung der Durchdringung zweier 3D-Objekte der virtuellen Welt zu verstehen. Prinzipiell können in Java3D beliebige Objekt-Objekt-Kollisionen zwischen allen 3D-Elementen festgestellt werden. Die in Java3D derzeit vorhandene Kollisionserkennung hat jedoch einige Nachteile, die sie momentan praktisch kaum einsetzbar erscheinen läßt:

1. Befindet sich ein Objekt bereits in einer Kollision, können keine weiteren Kollisionen mit anderen Objekten mehr festgestellt werden.
2. Da die Kollisionserkennung in einem eigenen Thread arbeitet, der parallel zum restlichen System ohne Synchronisationsmechanismen „läuft“, ist es möglich, daß Kollisionen so schnell auftreten und wieder aufgelöst werden, daß keinerlei Aktivierung der zugehörigen Kollisions-Behavior erfolgt.
3. Trotz Einsatz der zur Verfügung stehenden Möglichkeiten, die Geometrie für die Kollisionserkennung so einfach wie möglich zu gestalten, wird das Laufzeitverhalten der Applikation empfindlich verschlechtert.

Aufgrund dieser Nachteile eignet sich das Kollisionsystem von Java3D zur Zeit nicht für die Realisierung virtueller Welten.

Transparenz

Texturen mit einem ausreichend großen Transparenz-(Alpha-)Kanal erlauben die einfache Erzeugung von scheinbar realitätsnahen, feinen Strukturen (z.B. Drahtgitter vor einem Fenster). Würden diese Strukturen in einem Echtzeitsystem wie Java3D durch die Definition entsprechender Geometriedaten erzeugt, würde die Anzahl an Polygonen der Szene bei übermäßigem Einsatz deutlich ansteigen, was sich negativ auf das Laufzeitverhalten auswirken würde. Java3D besitzt nun das Problem, daß mehrere hintereinander liegende transparente 3D-Objekte nicht korrekt dargestellt werden: Transparente Objekte, die räumlich hinter anderen transparenten Objekten plaziert werden, erscheinen fälschlicherweise im Vordergrund. Ein derzeit bekannter Workaround löst dieses Problem nicht vollständig, da hierbei wiederum Probleme (additive Effekte) bei der Überschneidung transparenter Objekte auftreten.

Auf den ersten Blick mag die Lösung dieses Problems der Behebung einer fehlerhaften Softwareimplementierung gleichkommen. Der Grund, der jedoch eine schnelle Lösung seitens des Java3D-Entwicklungsteams verhindert, ist in einer der grundsätzlichen Entwicklungsziele des Java3D-APIs zu finden: Die Zustandsunabhängigkeit des Szenengraphen, d.h. verschiedene Teilbäume eines Szenengraphen beeinflussen sich nicht gegenseitig z.B. durch Transformationen. Durch diese Zustandsunabhängigkeit wird es dem Java3D-System, anders als z.B. in OpenInventor ermöglicht, den Szenengraphen in beliebiger Reihenfolge, beispielsweise von links nach rechts, aber auch parallel darzustellen, wodurch eine Verteilung des Darstellungsprozesses auf mehrere CPUs innerhalb eines Rechnersystems oder eine Aufteilung auf mehrere Rechner möglich wird. Die korrekte Darstellung transparenter Objekte bedarf jedoch eines gesonderten Darstellungsschrittes nach der Darstellung der undurchsichtigen Objekte, in dem die transparenten Objekte von hinten nach vorne vom Betrachterstandpunkt aus gezeichnet werden, damit das 3D-System eine realitätsnahe

Darstellung erzeugen kann. Diese Sortierung steht natürlich im Widerspruch zur Zustandsunabhängigkeit und damit zur gewünschten Verteilbarkeit des Darstellungsprozesses.

Zusammenfassung und Ausblick

Die Implementierung der virtuellen Mall in Java3D hat gezeigt, daß das Java3D-API für die Realisierung größerer 3D-Anwendungen geeignet ist. Es werden dem Entwickler umfangreiche Möglichkeiten geboten, die Ausführungsgeschwindigkeit und den Speicherverbrauch zu optimieren. Das vorgestellte Komponentensystem zeigt auf, in welcher Weise große Szenengraphen in Richtung Kapselung, Mehrfachverwendung und flexible Konfiguration zielgerichtet strukturiert werden können. Der Prototyp demonstriert weiterhin die gelungene Integration des Java3D-APIs in die Java-Plattform. Durch die Plattformunabhängigkeit (nur für Plattformen gegeben, für die eine Java3D-Implementierung vorliegt, ähnlich wie bei VRML-Browsern), wird ein Einsatz von hochperformanten Java3D-Anwendungen im Internet möglich. Die beschriebenen Probleme bzw. Einschränkungen sind, bis auf die Transparenz-Problematik, implementierungsbedingt und werden in zukünftigen Releases sicherlich behoben.

Literatur

1. Chris Wewerka: Aufbau einer virtuellen Mall mit Navigationselementen in Java3D, Diplomarbeit, März 2000, http://www.ThreeDimensions.de
2. Sun Microsystems: Java3D API Specification, Juni 1999
3. Henry Sowizral, David R. Nadeau: An Introduction to Programming AR an VR Applications in Java3D, 1999, http://www.sdsc.edu/~nadeau/Courses/VR99
4. Bernie Roehl, Justin Couch, Cindy Reed-Ballreich, Tim Rohaly, Geoff Brown: Late Night VRML 2.0 with Java, Ziff-Davis Press, 1997
5. Jörg Kloss et al.: VRML97 – Der neue Standard für interaktive 3D-Welten im WWW, Addison-Wesley, 1998
6. Michael Lorenz: Entwurf und Ausgestaltung eines virtuellen Shops mit Java3D, Diplomarbeit, September 1999
7. Web3D Consortium, http://www.web3d.org