4 Die haptische Rendering Pipeline

4.1 Grundlagen des Rendering in der Computergrafik

Beim konventionellen Rendern visueller Grafiken kann man heute unter einer Vielzahl von Verfahren wählen, die für die jeweilige Anwendung und unter den gegebenen technischen Rahmenbedingungen (insbesondere bezüglich der verfügbaren Rechenleistung) bestmögliche Ergebnisse liefern.

In diesem Unterkapitel wird zunächst kurz auf einige Fragen aus dem Fachgebiet „Computergrafik" eingegangen. Zu diesen Fragen zählen zum Beispiel solche wie

Was ist das „Modell" in diesem Zusammenhang?

Wo kommt es her?

Wie wird es verarbeitet?

Was ist das Ergebnis der Verarbeitung?

...

Diese und ähnliche Fragen lassen sich bis zu einer gewissen Tiefe beantworten, wenn man den folgenden Beispiel-Ablauf der Generierung einer Computergrafik (einschließlich des Rendering) verfolgt. Zur fundierteren und genaueren Auseinandersetzung mit dem Thema sei auf die Fachliteratur, insbesondere auf Lehrbücher (z.B.[FOLEY ET AL 90, HEARN & BAKER 94]) verwiesen. Hier findet sich auch die Vorlage für die Darstellung einer Rendering-Pipeline in Abbildung 4-1.

Je nach Rendering- Verfahren werden verschiedene Prozeßstufen durchlaufen. Am Beginn der Verfahren steht jeweils das Modell, und das Ergebnis ist eine auf einem Display angezeigte Grafik. Der Weg, der vom Modell zur Grafik führt, heißt *Rendering Pipeline*. Im einfachsten (wenn auch sehr rechenaufwendigen) Fall - beim ray tracing - sieht die visuelle Rendering Pipeline so aus:

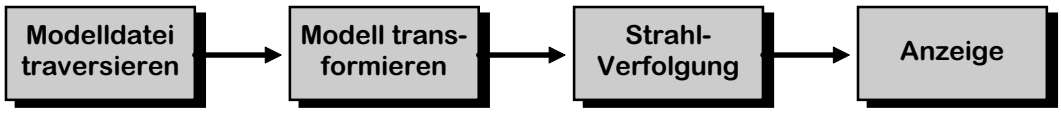


Abbildung 4-1 (Visuelle) Rendering pipeline für das *ray tracing* Verfahren (nach [FOLEY ET AL. 90, S.809])

Der gesamte Weg bis zum Bild (bzw. bis zur „Anzeige") läßt sich anhand folgender Schritte beschreiben:

1. Modellierung

Die darzustellende Szene wird „modelliert“. Dazu verwendet man heute in der Regel Programme (sogenannte „Modeller“), die eine direkte Erzeugung und Manipulation geometrischer Objekte am Computer ermöglichen.

Solche Modeller sind oft Teile größerer 3D-Programmpakete, die auch Renderer enthalten.

Man erkennt in der Abbildung einen relativ großen Bereich, in dem die Objekte in der Szene angeordnet werden können. Die logische Baumstruktur der Szene wird ebenfalls deutlich (unten rechts).

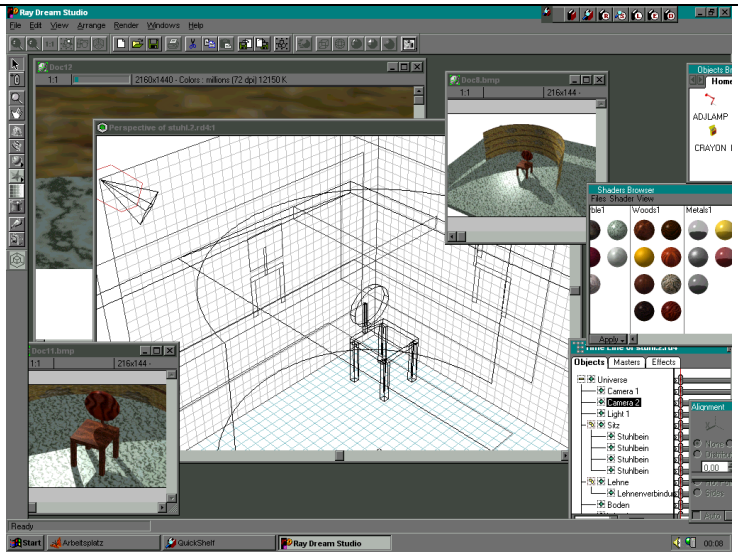


Abbildung 4-2 Bildschirm-Foto eines typischen Modellers (Ray Dream Designer™): Mit verschiedenen Werkzeugen (links oben) können Objekte erzeugt und manipuliert werden. Oberflächeneigenschaften stehen in Form einer Materialbibliothek (rechts) zur Verfügung. Kleine Vorschaubilder geben einen Eindruck vom zu erwartenden Ergebnis. Unten rechts das Hierarchie-Fenster, das die logisch Baumstruktur der Szene zeigt.

2. Modell

Ergebnis des Modellierens ist eine Datenstruktur, die alle Eigenschaften der Szene beschreibt (Objekte, Lichtquellen, Kameraposition und -orientierung, ...). Diese Rechner-interne Datenstruktur kann nun in eine Datei geschrieben werden. Hierzu existiert eine große Anzahl meist herstellerspezifischer Dateiformate. Als offenes textbasiertes (und daher auch von Menschen lesbares) Dateiformat ist die „*Virtual Reality Modelling Language*“ (VRML, [VRML SPEC 96]) sehr verbreitet.

```
#VRML V1.0 ascii
# Szene mit stuhl
#Created using Ray Dream Designer(R) 4
Separator {
        PerspectiveCamera {
                position 3.662 4.195 4.109
                orientation -.707 .707 0 .955
                heightAngle .927
        }
SpotLight {
                location -1.4 1.4 1.4
                direction .577 -.577 -.577
                cutOffAngle 1.571
                color .5 .5 .5
        }
Separator {
        Info { string "stuhl" }
        Separator {         #Sitz
                Info { string "sitzflaeche" }
                Translation { translation 0 110 0 }
                Cube { height 20 width 200 depth 200 }
                }
        Separator {
                Info { string "stuhlbein " }
                Translation { translation 90 0 90 }
                Cube { width 20 depth 20 height 200 }
                }
...
        Separator {         # stuhllehne
                Info { string "stuhllehne" }
                Translation { translation 0 130 80 }
                Info { string "lehnenbefestigung" }
                Cube { width 50 depth 20 height 20 }
                Translation { translation 0 5 30 }      #0 30 0
                Cube { width 10 height 10 depth 40 }
                Separator { Info { string "einstellschraube" }
                        Translation { translation 0 0 25 }
                        Rotation { rotation 1 0 0 1.57079 }
                        Cylinder { radius 10 height 10 }
                        }
...
```

Abbildung 4-3 VRML-Modell. Ein VRML-Modell kann auch ohne Modeller, nur mit einem Text-Editor, erstellt und bearbeitet werden. Aus dieser VRML-Datei wurden Kamera und Beleuchtung später entfernt, da sie für das haptische Rendering bestimmt ist.

3. Modelldatei traversieren

Der erste Schritt des eigentlichen „Rendering“ ist die Traversierung der Modelldatei. Dabei wird die Baumstruktur des Modells Knoten für Knoten durchgegangen. Die jeweiligen lokalen Transformationen (z.B. Rotationen und Translationen (Verschiebungen) von Objekten) werden dabei berücksichtigt.

4. Modell transformieren

Alle Koordinaten aus dem Modell werden aus ihrem lokalen Bezug herausgelöst und in sogenannte Weltkoordinaten umgerechnet. Die Modellkoordinaten werden dabei nicht verändert, sondern lediglich in ein anderes Bezugssystem überführt und schließlich per Projektion auf eine Fläche abgebildet. Auf die verschiedenen Zwischenschritte wie „*visible-surface determination*“, *clipping* und die unterschiedlichen Schattierungsverfahren soll hier nicht weiter eingegangen werden. Wegen des Ziels des visuellen Renderings (ein visuell glaubwürdiges Bild zu erzeugen) wird bei allen Zwischenschritten nach den Gesetzen der Optik und der visuellen Wahrnehmung vorgegangen.

5. Strahlverfolgung

Nun wird berechnet, welche Farbe jeder einzelne Punkt auf der Anzeige hat. Die Anzeige ist hier meist der Bildschirm, gedanklich liegt dem *ray tracing* allerdings die fotografische Kamera mit einer Filmebene zugrunde: Es wird berechnet, welchen Weg der Lichtstrahl genommen hat, der den betreffenden Punkt einfärbt.

Insbesondere Reflexionen bedeuten erneute Strahlverfolgungen. Prinzipiell müßte jeder Lichtstrahl bis zu seiner Lichtquelle verfolgt werden. In der Praxis geht man jedoch von einem gewissen Umgebungslicht aus, um die Anzahl der zu berechnenden Reflexionen gering zu halten.

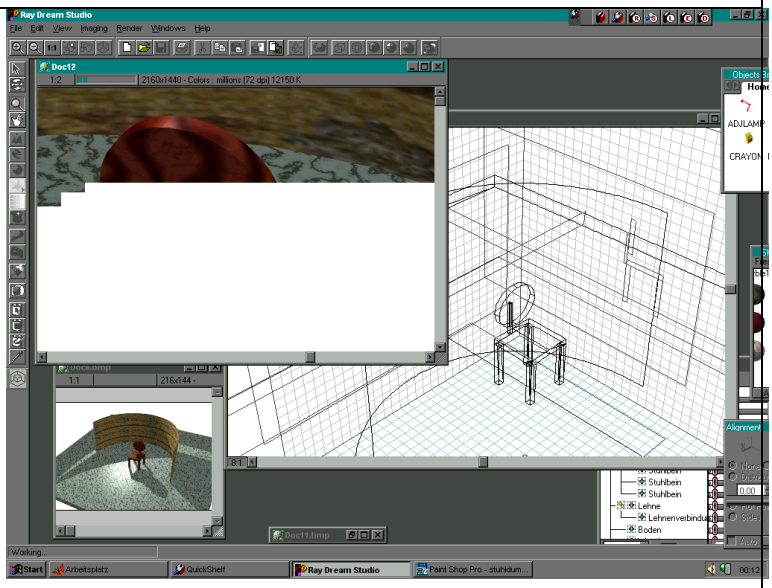


Abbildung 4-4 Bildschirm-Foto des Ray Dream Studio™ 3D-Programmpakets beim *ray tracing*. Im Fenster im Vordergrund ist gut zu erkennen, daß das Bild nicht Objekt für Objekt aufgebaut wird, sondern Pixel für Pixel. Welches Objekt für die Farbe eines Bildpunkts (Pixel) verantwortlich ist, ergibt sich aus der Strahl-Verfolgung eines gedachten Lichtstrahls.

Dazu verfolgt man den Lichtstrahl „rückwärts“ von seinem Endpunkt auf der gedachten Filmebene durch das optische System der Kamera zum ersten Objekt der Szene, das auf seinem Weg liegt. Aus der Farbe dieses Objekts (an der betreffenden Stelle) und dem darauf fallenden Licht sowie aus einem Anteil von Reflexion (Spiegelung) wird die Farbe des Punktes auf der Filmebene berechnet.

6. Anzeige

Das fertig gerenderte Bild enthält nun eine Darstellung der Szene, die mit ihren Verzerrungen, Verdeckungen, Schatten und Lichtreflexen einem Foto recht ähnlich ist. Das Maß dieser Ähnlichkeit wird oft als Maß der Qualität visueller Computergrafiken herangezogen. Dies zeigt die Bedeutung des Begriffs „Fotorealismus“ in der Computergrafik.

Abbildung 4-5 „fotorealistisch“ mit dem *ray tracing* Verfahren gerendertes Bild der Stuhlszene

Das *ray tracing* Verfahren orientiert sich möglichst nah an den physikalischen Abläufen bei der optischen Bildgenerierung durch das Fotografieren. Es liefert Ergebnisse (Bilder), die denen einer Kamera sehr ähnlich sind. Da sehende Menschen der westlichen Zivilisation Fotos als visuell glaubwürdige Bilder zu akzeptieren gelernt haben, werden auch fotorealistisch per *ray tracing* gerenderte Bilder als visuell plausibel empfunden.

Andere visuelle Rendering-Verfahren haben eine wesentlich längere Kette von Prozeßstufen, wobei die einzelnen Schritte jedoch weniger aufwendig zu berechnen sind als die „Strahlverfolgung“ beim *ray tracing*. Allen visuellen Rendering-Verfahren gemeinsam ist die Abbildung des in Weltkoordinaten umgerechneten 3D-Modells in 3D viewport-Koordinaten, die dann wiederum gerastert auf dem 2D-Display angezeigt werden können. Grundlage für diese Abbildung ist eine Szenenbeschreibung (Modelldatei) mit Lichtquellen und einem Kameramodell.

4.2 Schritte beim haptischen Rendering

In diesem Kapitel wird nun eine *haptische Rendering Pipeline* vorgestellt, und die ihr zugrunde liegenden Algorithmen und Datenstrukturen sollen skizziert werden.[20] Danach (im Kapitel 5 „**Vorgehen bei der räumlichen Darstellung**“) wird auf die einzelnen Verfahren, ihre technischen und perzeptiv-kognitiven Grundlagen und deren Anwendungen einzugehen sein.

Das haptische Rendering hat nicht eine visuelle, sondern eine haptisch glaubwürdige Abbildung zum Ziel. Daher kann (und muß) auf ein Kameramodell und auf eine Berücksichtigung von Lichtquellen verzichtet werden. Statt dessen müssen die mechanischen Eigenschaften, insbesondere die tastbaren, in das Bild einfließen. Während der erste Schritt, die Traversierung der Modelldatei mit anschließender Transformation der Modellkoordinaten in ein Weltkoordinatensystem noch mit dem Vorgehen beim visuellen Rendering vergleichbar ist, unterscheiden sich die darauf folgenden Schritte grundlegend vom Vorgehen beim visuellen Rendern.

Abbildung 4-6 zeigt einen Abriß der einzelnen Schritte beim haptischen Rendering.

[20] Haptisches Rendern ist hier immer die Abbildung auf eine Fläche zur taktilen *und* kinästhetischen Wahrnehmung. In [SABMSZ95] wird unter „*Haptic Rendering*“ die Abbildung in einen 3D-Raum zur rein kinästhetischen Wahrnehmung mit dem PHANToM verstanden.

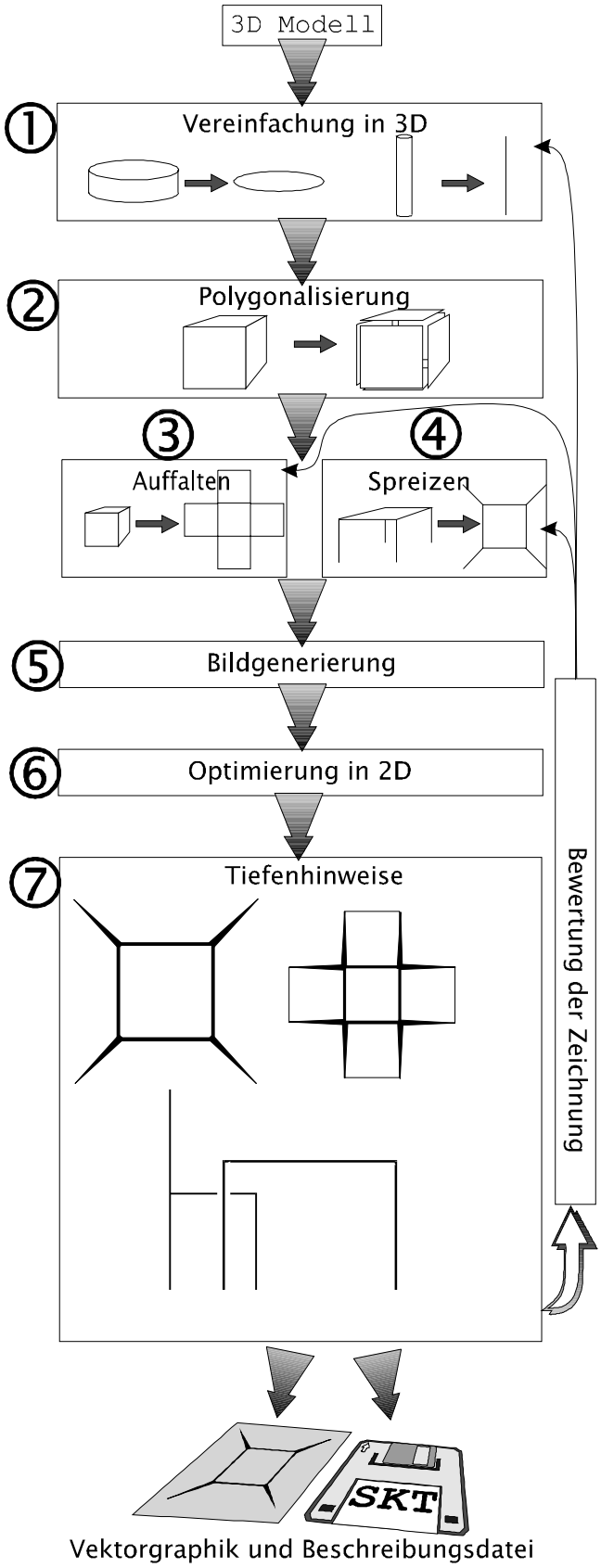


Abbildung 4-6 Die haptische Rendering-Pipeline: Aus einem 3D-VRML-Modell werden eine taktile Grafik und die zugehörige Beschreibung erzeugt (Details im Text).

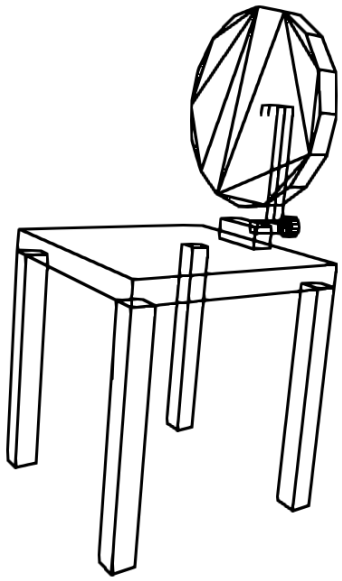
Vor dem eigentlichen Rendering wird eine Modelldatei der Szene mit räumlichen Koordinaten der einzelnen Objekte in eine interne Datenstruktur eingelesen. Grundsätzlich eignen sich alle Modellbeschreibungssprachen, die polygonbasiert sind oder auf geometrischen

Primitiven beruhen, die sich polygonalisieren lassen (Zylinder, Kugeln, Quader, usw.). In der realisierten Implementierung wurden Modelle im VRML-Format verwendet [VRML SPEC 96]. Dieses Format bietet den zusätzlichen Vorteil, daß die Objekte benannt werden können (siehe Abbildung 4-7).

```
#VRML V1.0 ascii

# Szene mit stuhl

Separator {
      Info {string "stuhl" }
      Separator { #Sitz
            Info {string "sitzflaeche" }
            Translation {translation 0 110 0 }
            Cube {height 20 width 200 depth 200 }
            }
      Separator {
            Info {string "stuhlbein " }
            Translation {translation 90 0 90 }
            Cube {width 20 depth 20 height 200 }
            }
      Separator {
            Info {string "stuhlbein " }
            Translation {translation -90 0 -90 }
            Cube {width 20 depth 20 height 200 }
            }
      Separator {
            Info {string "stuhlbein " }
            Translation {translation 90 0 -90 }
            Cube {width 20 depth 20 height 200 }
            }
      Separator { # stuhlbein4
            Info {string "stuhlbein " }
            Translation {translation -90 0 90 }
            Cube {width 20 depth 20 height 200 }
            }
      Separator { # stuhllehne
            Info {string "stuhllehne" }
            Translation {translation 0 130 80 }
            Info {string "lehnenbefestigung" }
            Cube {width 50 depth 20 height 20 }
            Translation {translation 0 5 30 }
            Cube {width 10 height 10 depth 40 }
            Separator {Info {string "einstellschraube" }
                  Translation {translation 0 0 25 }
                  Rotation {rotation 1 0 0 1.57079 }
                  Cylinder {radius 10 height 10 }
                  }
            Translation {translation 0 60 5 }
             Info {string " lehnenbefestigung" }
             Cube {width 10 height 110 depth 10 }
             Translation {translation 0 50 -15 }
             Cube {width 10 height 10 depth 20 }
            Translation {translation 0 0 -20 }
            Rotation {rotation 1 0 0 1.57079 }
             Info {string "lehne" }
             Cylinder {radius 90 height 20 }
            }
      }           #ende der szene
```



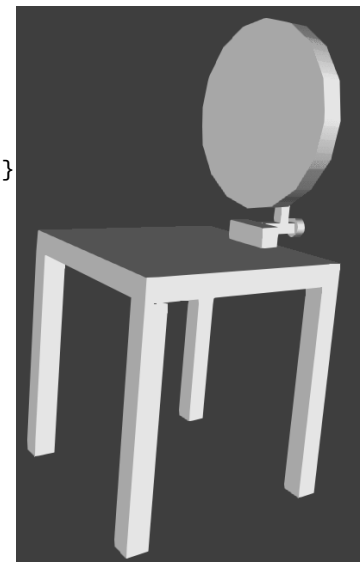


Abbildung 4-7 Eine einfache Modellbeschreibung im VRML-Format. Das Modell enthält die Objekt-Namen als `Info { string "name"}.`

4.3 Die interne Datenstruktur: der Modellbaum

Durch die strukturierte Form der Modellbeschreibung ergibt sich auch eine strukturierte Modellrepräsentation im Renderer. Diese Struktur ist im mathematischen Sinn ein Baum, dessen Wurzel die ganze Szene ist. In der VRML-Spezifikation wird hier von einem Szenengraphen (*scene graph*) gesprochen [VRML SPEC 96]. Dieser Szenengraph definiert eine Ordnung seiner Knoten: Jeder Knoten beeinflußt jeden seiner Nachfolger. Ein Knoten kann Objekte verschiedener Art enthalten: Neben den eigentlichen Gegenständen (geometrische Primitive oder zusammengesetzte Objekte) sind unter anderem auch Transformationen und Informationsknoten möglich.

Die Identifizierung der „Kind"-Objekte geschieht im Falle von VRML durch das „Separator"-Konstrukt: Alle Objekte innerhalb der Separator-Klammern werden als Kinder dieses Objekts aufgefaßt. Sein Name ergibt sich aus dem Info-Feld unmittelbar hinter dem Separator. Die Kind-Objekte können selbst wieder strukturiert sein, so daß sich die Baumstruktur rekursiv fortsetzt. Blätter (Endknoten) des Baums sind zunächst die geometrischen Primitive, die im VRML-Modell auftauchen: Quader, Zylinder, Polygone usw.

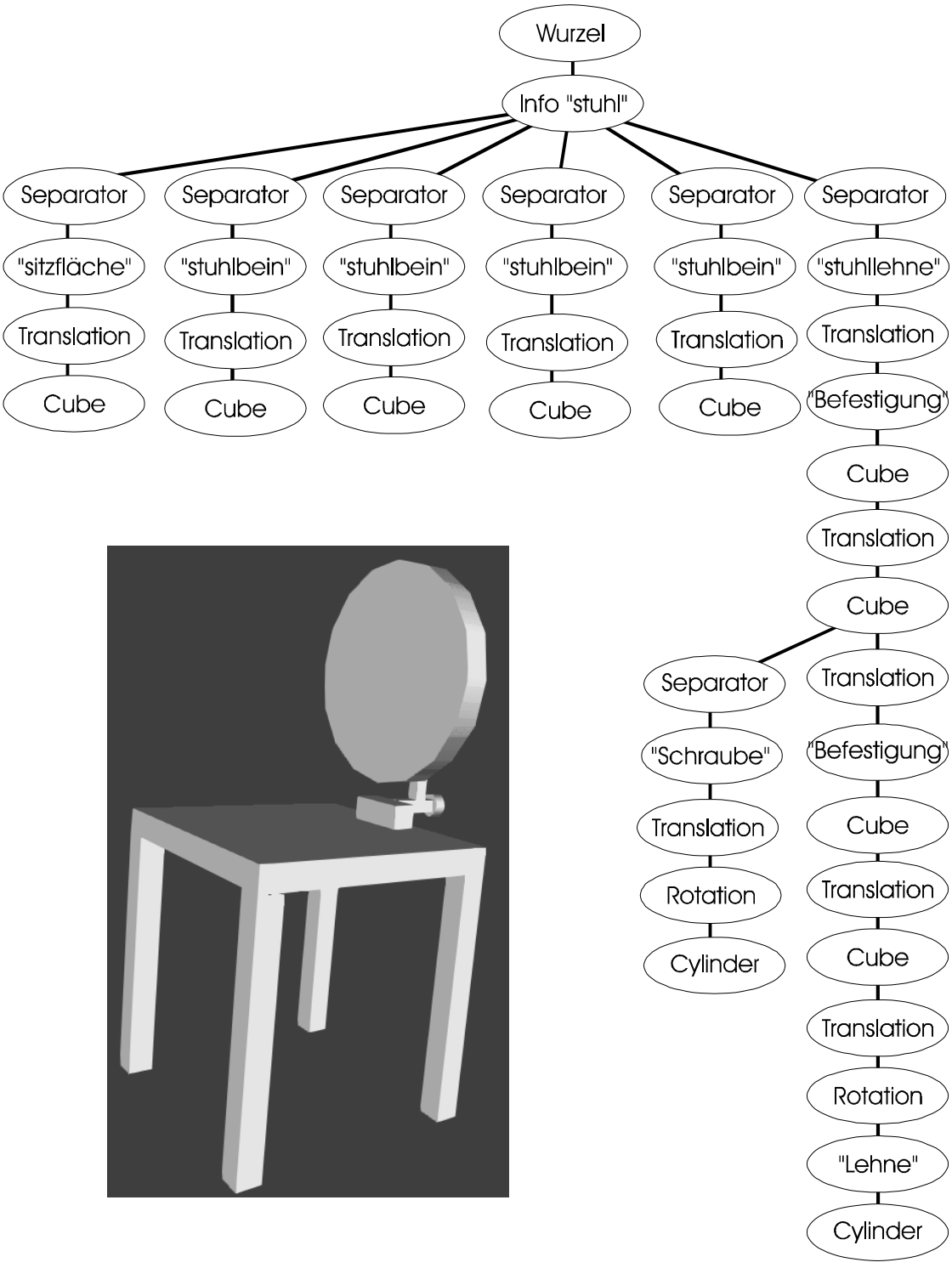


Abbildung 4-8 Diagramm des Modellbaums, der aus dem VRML-Modell in Abbildung 4-7 generiert wurde.

Die interne Repräsentation des Modells als aus dem Szenengraph abgeleiteter Baum ermöglicht die Manipulation der räumlichen Struktur des Modells durch das Einfügen neuer Knoten in den Baum. Die ursprünglichen Knoten können dabei im wesentlichen unverändert bleiben und werden nur in einen anderen Kontext gesetzt, wodurch einerseits die Rücknahme der Manipulationen und andererseits das herkömmliche (visuelle) Rendern der Szene erleichtert wird.

Das in der Baumstruktur repräsentierte Modell der Szene durchläuft nun die haptische Rendering Pipeline, wobei es gegebenenfalls verändert und um (innere wie äußere) Knoten ergänzt wird. Der folgende Abschnitt skizziert diesen Vorgang.

4.4 Abschnitte der Rendering-Pipeline

Die in Abbildung 4-6 dargestellte Rendering Pipeline besteht aus 7 Stufen, von denen 2 (die Stufen ③ und ④) alternativ durchlaufen werden. Im Einzelnen werden die in den folgenden Abschnitten erläuterten Prozeßstufen durchlaufen. Die Bewertung der 2D-Darstellung (siehe auch Abschnitt 5.5) wirkt gegebenenfalls auf die räumlichen Manipulationen des Modells zurück.

Vereinfachung in 3D

Abschnitt 5.2 beschreibt die Verfahren zur Vereinfachung der räumlichen Szene im Detail. Prinzipiell werden (je nach Konfiguration) längliche Gegenstände durch Linien ersetzt und flache durch Polygone. Dadurch wird die Komplexität der Szene reduziert, insbesondere reduziert sich die Zahl der möglichen Faltungs- und Klapp-Operationen bei der Transformation in die Ebene.

Außerdem wird erst durch die Reduzierung länglicher Objekte auf Linien die Spreizung dieser Linien ermöglicht (siehe Abschnitt 5.3.3).

Polygonalisierung

Zu Vorbereitung auf das anschließende Auffalten müssen Objekte wie Zylinder und Quader polygonalisiert, d.h. in Polygone zerlegt, werden. In jedem Fall bleibt die Information erhalten, daß es sich bei dem nun polygonalisierten Objekt um ein zerlegtes komplexeres Gesamtobjekt handelt. Dadurch kann diese Information prinzipiell auch an den endgültigen Benutzer, den blinden Erfasser, ausgegeben werden.

Das Modell hat nun als Endknoten (Blätter) der Baumstruktur nur noch zweidimensionale Objekte: Polygone und Linien.

Transformation in die Ebene

Die Polygone und Linien haben bisher noch verschiedene Ausrichtungen im 3D-Raum. Sie können und müssen nun durch Rotationen, die als Transformationsobjekte in den Modellbaum eingefügt werden, parallel zur Darstellungsebene gebracht werden. Welche Flächen und Linien wie rotiert werden und was sich aus diesen Rotationen für Konsequenzen für Objekte, die topologisch mit soeben rotierten Gegenständen verbundenen sind, ergeben, wird im Abschnitt 5.3.2 näher behandelt.

Bildsynthese

Die Bildsynthese bedient sich im wesentlichen eines als „painters-algorithm“ bekannten Verfahrens: die nunmehr parallel zur Bildebene liegenden Einzelobjekte werden „von hinten nach vorn“ gezeichnet. Dadurch überlagern näher am Erfasser liegende Gegenstände solche, die weiter entfernt liegen. Diese Überlagerung kann geeignet hervorgehoben und als Tiefenhinweis genutzt.

Bei der Ermittlung der zu zeichnenden Linien und Flächen ist zu jedem End- oder Eckpunkt bekannt, zu welchem Objekt er gehört, da diese Information unmittelbar aus dem Modellbaum gewonnen wird. So kann gleichzeitig mit der Generierung der 2D-Zeichnungskoordinaten die Beschreibung der Szene erstellt werden (siehe 4.5).

Optimierung in 2D

Je nach Komplexität und Struktur der Szene kann es vorkommen, daß in der 2D-Repräsentation zwei Linien sehr dicht beieinander liegen. Dadurch kann es zu Problemen beim tastenden Verfolgen einzelner Linien kommen. Um diese Probleme vorherzusehen und möglichst zu vermeiden, kann ein 2D-Objekt mit solchen Linien gegebenenfalls verschoben werden, um den Linienabstand zu vergrößern. Dabei ändert sich allerdings die dargestellte Information: Sie beruht nun nicht mehr ausschließlich auf der ursprünglichen 3D-Szene. Daher ist statt einer automatischen Manipulation auch die Generierung angemessener Warnungen vorstellbar.

Tiefenhinweise

Die 2D-Zeichnung besteht nun aus einer Sammlung von Linien, die aus parallel zur Zeichnungsfläche liegenden Polygonen und Linien gewonnen wurden. Da die ursprünglichen Koordinaten der Endpunkte aller Linien bekannt sind, können nun die im Abschnitt 5.4 näher beschriebenen Verfahren zur Repräsentation räumlicher Tiefe angewendet werden: Liegt ein Endpunkt einer Linie „tiefer im Raum“ als der andere, kann die Linie als Keil dargestellt werden. Liegt eine Linie (Kante) eines geklappten (rotierten) Polygons im ursprünglichen Modell weiter hinten als die andere, kann sie dünner als diese gezeichnet werden.

Ebenso kann beim Zeichnen der Linien in der durch den „painters-algorithm“ festgelegten Reihenfolge jede Linie zunächst sehr breit in weiß und anschließend in Originalbreite in schwarz darüber gezeichnet werden. Dadurch wird der räumliche Eindruck erzeugt, daß sich überkreuzende Linien voreinander liegen. In **Tabelle 3-2** wurde bereits auf diesen Effekt hingewiesen.

4.5 Rendering-Resultate: Grafik und Beschreibung

Am Ende der Rendering Pipeline steht eine Vektorgrafik und eine zugehörige Beschreibungsdatei.

Die Vektorgrafik liegt zunächst als 2D-Datenstruktur im Rechner vor und kann nun problemlos gespeichert, elektronisch transportiert und beliebig oft ausgedruckt werden. Steht ein geeignetes Medium zur Verfügung, kann die Grafik direkt als haptische Zeichnung ausgegeben und erkundet werden.

Parallel zur Zeichnung wurde beim Rendering eine formale Beschreibung der Zeichnung (nicht der Szene) generiert, die sich auf die Namen der Objekte, deren Linien in der Zeichnung stehen, bezieht. Als Format dieser Beschreibung, die im wesentlichen aus den Linien- und Polygon-Koordinaten und den Namen der durch sie repräsentierten Objekte besteht, wurde das SKT-Format des AudioTouch-Systems verwendet (siehe [AUDIOTOUCH 95]).

Die Grafiken und die Beschreibung sind nun unmittelbar mit AudioTouch zu verwenden und zu erkunden. Außerdem können sie mit erkundungsunterstützenden Systemen wie TGuide (siehe Abschnitt 7.4) untersucht und erfaßt werden. Auf die Bedeutung der textuellen Information bei der Erkundung der Grafik wird im Abschnitt 5.7 näher einzugehen sein.

Durch die Erhaltung und Bereitstellung der Bedeutungsinformation aus dem Modell in der Grafik ergeben sich Ähnlichkeiten mit dem Hyper-Renderer von Emhardt [EMHARDT & STROTHOTTE 92]. Bedeutende Unterschiede sind vor allem die Unabhängigkeit des hier beschriebenen Ansatzes von Pixel-Rastern, wie sie beim herkömmlichen Rendern und auch

vom Hyper-Renderer benötigt werden und die Verwendung der Objektnamen als Indizes für weitere Informationen, die in einer separaten Datenbank verwaltet werden kann.

Nach dieser knappen technischen Beschreibung der haptischen Rendering Pipeline soll im folgenden Kapitel auf Rahmenbedingungen, Verfahren und die Modellierung der kognitiven Belastung im Zusammenhang mit haptischen Zeichnungen räumlicher Gegenstände eingegangen werden. Dabei wird oft Bezug auf die im vorliegenden Kapitel 4 beschriebenen Algorithmen und Datenstrukturen genommen werden.