Thema:

Dateiformate für dreidimensionale Daten

Ausarbeitung im Rahmen des Seminars „Unterstützung von Landminendetektion durch Bildauswertungsverfahren und Robotereinsatz“

im Fachgebiet Wirtschaftsinformatik
am Institut für Informatik - Fachbereich Mathematik und Informatik

Themensteller: Dr. Dietmar Lammers
Betreuer: Dipl.-Inform. Steffen Wachenfeld

vorgelegt von: Kolja Thierfelder
Steinfurter Str. 128a
48149 Münster
Tel.: 0251-2007741
thierfel@uni-muenster.de

Abgabetermin: 2004-01-05

Inhaltsverzeichnis

1 Einführung

Jedes Bildverarbeitungsprogramm erlaubt in der Regel das Laden und Speichern einer ganzen Reihe von Dateiformaten. Damit wird dem Anwender die Möglichkeit gegeben, abhängig von seinen weiteren Plänen das für ihn sinnvollste Format zu verwenden. Er kann ein Bildformat danach wählen, ob Plattenspeicher gespart werden soll, ob das Bild mehr oder weniger schnell aufgebaut werden muss, ob das Bild in einem Desktop-Publishing-Programm verwendet wird oder ob es an einen professionellen Belichtungsservice gehen soll.

Dateiformate, die sich für den Umgang mit dreidimensionalen Daten eignen (im Folgenden als dreidimensionale Dateiformate oder 3D-Formate bezeichnet), speichern Beschreibungen von Form und Farbe künstlich erzeugter 3D-Modelle und Objekten der realen Welt. Modelle der dreidimensionalen Objektrepräsentation bestehen häufig aus Polygonen und glatten Oberflächen, die ihrerseits mit Beschreibungen bestimmter Eigenschaften wie Farbe, Texturen, Reflektionen etc. kombiniert werden. Zunehmend an Bedeutung gewinnen daneben Ansätze der Volumengrafik, bei denen Objekte in kleine Volumenpixel (Voxel) zerlegt werden und jedem Voxel Eigenschaften wie Farbe oder Transparenz zugeordnet werden.

Ziel dieser Ausarbeitung ist es, einen Überblick über den derzeitigen Stand zur computerunterstützten Repräsentation dreidimensionaler Objekte zu geben. Dazu werden in Kapitel 2 wichtige Grundlagen der dreidimensionalen Objekt- und Szenenbeschreibung behandelt, um zu zeigen, welche Möglichkeiten es grundsätzlich gibt, um dreidimensionale Modelle abzuspeichern. Das dritte Kapitel nimmt eine Klassifizierung gebräuchlicher 3D-Formate vor und befasst sich exemplarisch mit den Formaten DXF, OBJ, 3DS, VRML und X3D. Ein kurzes Fazit wird in Kapitel 4 gezogen.

2 Dreidimensionale Objektrepräsentation

Die Beschreibung eines Objekts ist komplexes und viel diskutiertes Problem in der Computergrafik.[1] Auch im 3D-Bereich haben die verschiedenen Repräsentationsformen ihre Vor- und Nachteile; für die vielen bestehenden Probleme gibt es keine universelle Lösung.

Die geometrischen Daten der in 3D-Formaten abzuspeichernden Objekte müssen in geeigneten Datenstrukturen abgelegt werden. Die Wahl einer geeigneten Objektrepräsentation ist keine einfache Aufgabe. Die Darstellungsform wird sowohl durch die Rendering-Technik, d.h. die Art der Erzeugung eines Bildes, als auch durch die Anwendung bestimmt. Eine Vielzahl von Aspekten können bzw. müssen berücksichtigt werden. Zu nennen sind z.B. geringer Speicherbedarf, schnelle Darstellbarkeit, die Durchführbarkeit linearer Transformationen und Kombinationsoperatoren, die Möglichkeit der Interaktion und die Möglichkeit der automatischen Generierung der Datenstruktur anhand von digitalisierten realen Objekten (z.B. mittels 3D-Scanner oder CT).[2]

Im Folgenden werden einige der verbreitetsten Repräsentationsschemata vorgestellt: Die Punktwolke, das Drahtmodell, Flächen- und Volumenmodelle.

2.1 Punktwolke

Die Punktwolke[3] (*Scatter Plot*) ist die einfachste Repräsentation dreidimensionaler Daten. Die Darstellung des Objektes erfolgt durch eine Menge von Punkten im Raum. Es bestehen keine Informationen über Kanten, Volumen und Oberfläche der darzustellenden Objekte. Dennoch kann aus einer Punktwolke heuristisch eine Objektoberfläche ermittelt werden, indem z.B. Polygone zwischen den jeweils am nahesten benachbarten Punkten gespannt werden. Dieses Vorgehen führt jedoch oft nicht zum gewünschten Ergebnis.

Die Datenstruktur einer Punktwolke ist sehr einfach: Die Elementarobjekte sind die Punkte, und ein Objekt besteht aus einer Menge von Punkten, die in einer Liste gespeichert werden:

[1] Vgl. Watt (2002), S. 43.

```
Objekt    =    Liste von Punkten
Punkt     =    3 Koordinaten (x,y,z) im Raum
```

Bei Punktwolken ist der Interpretationsspielraum des Betrachters in der Regel sehr hoch. Um gekrümmte Kanten und Oberflächen zu repräsentieren, benötigt man sehr viele Punkte, was zu einem entsprechend hohen Speicherplatzverbrauch führt.

Die Vorteile der Punktwolke liegen – bei angemessener Verwendung – im geringen Speicherplatz und schnellen und einfachen Transformationen. Zudem ist eine Modellgenerierung durch Digitalisierung möglich.

2.2 Drahtmodell

Beim Drahtmodell (*Wireframe Model*) beschränkt sich die rechnerinterne Repräsentation auf Kanten, die zwischen bestimmten Punkten verlaufen. In der Literatur[4] wird häufig davon ausgegangen, dass sich ein Drahtmodell ausschließlich aus geraden Kanten zusammensetzt. Andere Autoren[5] lassen auch Bögen, Kreise etc. zu. Hier soll der einfachere Fall der Beschränkung auf gerade Kanten betrachtet werden. Eine übliche hierarchische Organisation der Daten ist folgende:

```
Objekt        =   Liste von Verweisen auf Kanten
Kantenliste   =   Liste aller Kanten im Modell
Kante         =   Zwei Verweise auf Punkte
Punkteliste   =   Liste aller Punkte im Modell
Punkt         =   Drei Koordinaten (x,y,z) im Raum
```

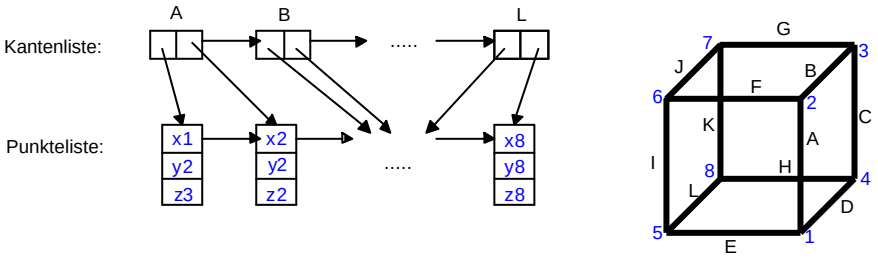

Abbildung 1: Datenstruktur des Drahtmodells

---

[2] Vgl. Gröller, Theußl (2002), S. 1.
[3] Vgl. z.B. Gröller, Theußl (2002), S. 1-2.
[4] Vgl. Gröller, Theußl (2002), S. 2-3.
[5] Vgl. Pfund (1999), S.2.

Das Drahtmodell liefert mehr Informationen über ein Objekt als die Punkwolke. Da jedoch auch in diesem Modell keine Informationen über die Flächen verwaltet werden, liegt die Interpretation der dargestellten Linien beim Benutzer, was unter Umständen einiges an dreidimensionalem Vorstellungsvermögen voraussetzt und nicht zu eindeutigen Ergebnissen führt. Es ist nicht möglich, festzustellen, welche Linien von einer Fläche verdeckt werden, also nicht sichtbar sind. Ein weiterer Nachteil ist der hohe Speicherbedarf bei krummen Objekten.

2.3 Flächenmodell

Um dreidimensionale Objekte realistischer und in richtiger Sichtbarkeit darstellen zu können, braucht man zumindest die Information, zwischen welchen Kanten sich Flächen befinden. Meistens wird dabei so vorgegangen, dass jedes Objekt direkt aus den das Objekt begrenzenden Flächen konstruiert wird; man spricht daher von einem Flächenmodell.[6]

Zur Modellierung der Flächen können verschiedene Grundelemente wie Polygone, Kreis- oder Ellipsenflächen oder mathematisch definierte Flächen verwendet werden. Flächenmodelle werden häufig dort eingesetzt, wo komplexe, mehrfach gekrümmte Oberflächen modelliert und bearbeitet werden sollen, wie im Flugzeug- und Automobilbau.

Da beim Flächenmodell die Objekte nur aus Oberflächen bestehen und daher kein Inneres haben, treten verschiedene Probleme auf, z. B. bei der Durchführung von Schnitten. Teilweise transparente Objekte ohne definierte Oberflächen wie Wolken, Explosionen oder Feuer sind damit nur über Tricks simulierbar. In diesem Bereich kann ein anderes Prinzip, die Volumengrafik, seine Stärken ausspielen (vgl. Kapitel 2.4).

Im Folgen werden die beiden am weitesten verbreiteten Flächenmodelle, Polygonnetze und bikubische Parameteroberflächen, vorgestellt.

[6] Vgl. Gröller, Purgathofer (1999), S. 812.

Oberflächenbeschreibung durch Polygonnetze

Das am häufigsten anzutreffende Flächenmodell zur Repräsentation dreidimensionaler Objekte ist das Polygonnetz.[7] Die Oberflächen setzen sich dabei aus einer Menge von planaren Polygonen zusammen. Grundsätzlich ist jede Art von Polygonen denkbar; in der Praxis werden jedoch meist nur Dreiecke oder Rechtecke verwendet. Die Datenstruktur des Polygonnetz-Modells kann wie folgt aussehen:

```
Objekt       =   Liste von Verweisen auf Polygone
Polygonliste =   Liste aller Polygone im Modell
Polygon      =   Liste von Verweisen auf Kanten
Kantenliste  =   Liste aller Kanten im Modell
Kante        =   Zwei Verweise auf Punkte
Punkteliste  =   Liste aller Punkte im Modell
Punkt        =   Drei Koordinaten (x,y,z) im Raum
```

Konsistenz- und Integritätstests, z.B. bezüglich Geschlossenheit und Planarität, sind leicht durchführbar. Es hat sich auch gezeigt, dass die einzelnen Einheiten günstig für den Rendering-Prozess sind. Allerdings braucht man dabei in jedem Fall sogenannte Hidden-Line- oder Hidden-Surface-Verfahren, die ermitteln, welche Teile sichtbar sind und welche nicht. Es ist schwierig, komplexe Formveränderungen vorzunehmen. Wenn ein Objekt einmal erstellt wurde, kann kein Polygon geändert werden, ohne dass auch seine Nachbarn verändert werden. Wie beim Drahtmodell ist auch beim Polygonnetz die Darstellung gekrümmter Oberflächen problematisch. Sie müssen durch planare Polygonflächen approximiert werden, was bei guter Näherung zu einer unvertretbar hohen Zahl von Polygonen führen kann.

Trotz seiner Nachteile dominiert das Polygonnetz-Modell die übliche Computergrafik, was neben der einfachen Erstellung von Polygon-Objekten an der Entwicklung effizienter Algorithmen und Hardware zum Rendern dieser Darstellung liegt.[8]

Polygonnetze sind eher eine Maschinendarstellung als eine brauchbare Benutzerdarstellung und dienen in dieser Eigenschaft oft anderen Darstellungen, die nicht direkt gerendert werden können. So werden bikubische Parameteroberflächen, CSG- und Voxel-Darstellungen oft in Polygonnetze umgewandelt, um sie dann zu rendern.

Bikubische Parameteroberflächen

Im Unterschied zur Darstellung durch Polygonnetze erfolgt die Abbildung eines dreidimensionalen Objekts bei den *bikubischen Parameteroberflächen* durch gekrümmte Oberflächen, die als *Patches* bezeichnet werden.[9]

Jedes Patch ist durch mathematische Formeln festgelegt, welche die Position im dreidimensionalen Raum sowie die Form des Patches bestimmen. Die Formeln ermöglichen es, jeden Punkt auf der Oberfläche eines Patches zu erzeugen. Form oder Krümmung eines Patches können durch eine Bearbeitung der mathematischen Beschreibung modifiziert werden, woraus sich weitreichende interaktive Möglichkeiten ergeben.

Dennoch gibt es auch bei den bikubischen Parameteroberflächen bedeutende Probleme. Es ist sehr aufwändig, die Patches direkt zu rendern oder zu visualisieren. Wenn die Form eines einzelnen Patches in einem Netz von Patches verändert wird, entstehen Probleme, die „glatten Übergänge" zwischen dem Patch und seinen Nachbarn zu bewahren.

Bikubische parametrische Patches können entweder eine exakte oder eine angenäherte Darstellung sein. Sie können nur eine exakte Darstellung von sich selbst sein, was bedeutet, dass jedes Objekt – z.B. ein Karosserieblech – nur dann genau dargestellt werden kann, wenn seine Form exakt der Form des Patches entspricht.

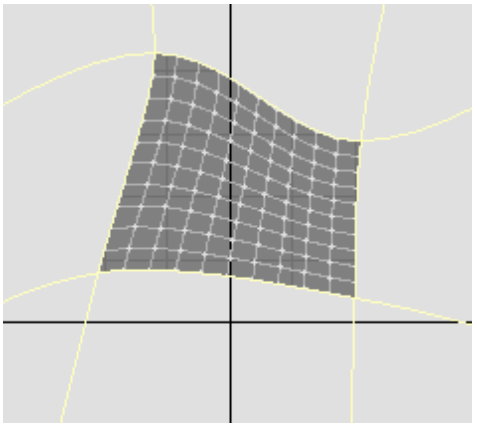

Abbildung 2: Bikubisches Patch[10]

[7] Vgl. Bender, Brill (2003), S. 191ff.
[8] Vgl. Watt (2002), S. 44.
[9] Vgl. Watt (2002), S. 83ff.

2.4 Volumenmodell

Das Volumenmodell geht von der Verwendung voller Körper aus und ist somit die natürlichste Methode der Modelldarstellung.[11] Das Volumen der Objekte wird direkt in der Datenstruktur repräsentiert, so dass es für im Volumenmodell erstellte Körper kein Problem ist, die richtige Sichtbarkeit zu ermitteln.

Mit dem Volumenmodell können auch 3D-Objekte beschrieben werden, die keine explizite Oberfläche aufweisen. Dies sind zum Beispiel Dichtefelder und 3D-Gitteranordnungen von Absorptionskoeffizienten, wie sie bei medizinischen Daten auftreten.[12]

Bei der Modellierung wird jeder Gegenstand entweder aus einfachen Elementarobjekten wie Würfel, Kugel, Oktaeder, Pyramide, Kegel usw. aber auch aus komplexeren Elementarobjekten wie Spurkörper, Freiformkörper, Fraktale usw. zusammengesetzt.

Beim Volumenmodell handelt es sich um eine eindeutige Darstellungsform, bei der - im Gegensatz zu Draht- und Flächenmodell - die Objekte immer konsistent sind. Da die Objekte immer volle Körper sind, treten auch beim Schneiden keine Probleme auf.

Der Ansatz der Volumengrafik hat eine große Bedeutung in der Medizin und in nahezu allen ingenieurstechnischen Disziplinen wie Maschinen- und Fahrzeugbau, Strömungstechnik, aber auch Bergbau oder Chemie. Im Folgenden werden die Volumenmodelle Voxel-Modell, Oktalbäume, Szenegraphen und Constructive Solid Geometry beschrieben.

Voxel-Modell

Beim Voxel-Modell wird eine dreidimensionale begrenzte Szene in zahlreiche geometrische Informationseinheiten von gleichen Ausmaßen unterteilt. Diese Informationseinheiten werden als Voxel bezeichnet und haben in aller Regel die Form von Würfeln (vgl. Abbildung 3).[13] Entsprechend der geforderten Genauigkeit der Szenenbeschreibung erfolgt die Wahl der Voxelanzahl. Eine hohe gewünschte Genauigkeit erfordert

---

[10] Vgl. Gilchrist (2001).
[11] Vgl. Purgathofer, Gröller (1999), S. 812.
[12] Der Absorptionskoeffizient beziffert den Anteil des Lichtes, welcher pro Einheit der Distanz in einem teiltransparenten Medium absorbiert wird.

eine hohe Voxelanzahl. Im Prinzip sind Voxel das dreidimensionale Analogum zu Pixeln eines zweidimensionalen Bildes.[14]

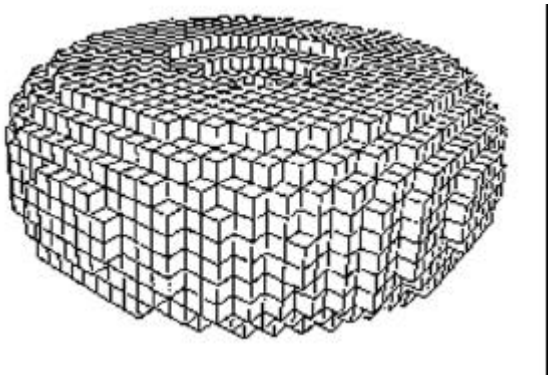

Abbildung 3: Voxel-Modell[15]

Jeder Voxel besitzt einen bestimmten Zustand. Dieser wird beim Erstellen der Szene ermittelt und kann später durch Werkzeuge zur Szenenmanipulation verändert werden. In der Regel wird als Zustand gespeichert, ob dieser Voxel leer oder ausgefüllt ist und welche Farbe die Füllmasse besitzt. Eventuell können, sofern bekannt, noch weitere Lichtcharakteristika des im Voxel befindlichen Materials gespeichert werden, z.B. stark oder schwach reflektierend, matt oder glänzend, klar oder diffus bei lichtdurchlässigem Material. Die Position des Voxels innerhalb der Darstellung wird meistens durch ein hierarchisches Verfahren (z.B. Oktalbaum, s.u.) bestimmt.

Der Speicherverbrauch einer Szene, die durch ein Voxel-Modell abgebildet werden soll, lässt sich wie folgt berechnen:

$$Speicherbedarf \; ? \; \frac{Kantenlänge\_Szene^3 \; ? Speicher\_pro\_Voxel}{Kantenlänge\_Voxel^3}$$

Der erforderliche Speicherplatz nimmt also mit der dritten Potenz zu, wenn die Genauigkeit erhöht (also die Kantenlänge eines Voxels verkleinert) wird. Aufgrund des hohen Speicherverbrauchs gehört die Voxel-Darstellung in nicht-wissenschaftlichen Bereichen nicht zu den bevorzugten Methoden. Sie wird dennoch verwendet, weil entweder die Rohdaten bereits in dieser Form vorliegen, weil es am einfachsten ist, die Daten in diese

[13] Zum Begriff des Voxel vgl. Häuser (1998), S. 6f.
[14] Vgl. Bender, Brill (2003), S. 373.
[15] Vgl. Hughes (1997)

Darstellung umzuwandeln (z.B. in der medizinischen Bildverarbeitung), oder aufgrund der Anforderungen eines Algorithmus.

Voxel können als Zwischendarstellung angesehen werden, vor allem in der medizinischen Bildverarbeitung, wo sie zweidimensionale Rohdaten mit der Visualisierung dreidimensionaler Strukturen verknüpfen. Alternativ können die Rohdaten selbst aus Voxeln bestehen. Dies ist bei vielen mathematischen Modellierungsschemata für dreidimensionale physikalische Phänomene wie die Strömungsdynamik der Fall.

Eine gebräuchliche Art, die Voxel-Daten zu organisieren, ist die Benutzung eines Oktalbaumes – einer hierarchischen Datenstruktur, die beschreibt, wie die Objekte einer Szene in dem von ihr eingenommenen dreidimensionalen Raum verteilt sind.

Oktalbäume

Die grundlegende Idee der Oktalbäume (*Octrees*)[16] ist in Abbildung 4 dargestellt. Zunächst wird ein ausreichend großer, die gesamte Szene umschließender Würfel gewählt. Dieser Würfel wird solange rekursiv in acht Teilwürfel halber Kantenlänge unterteilt, bis entweder der jeweilige Würfel ganz innerhalb oder ganz außerhalb des darzustellenden Körpers liegt bzw. bis eine vorgegebene Genauigkeit für die feinste Kantenlänge erreicht ist. Größere Bereiche, die eine einheitliche Information enthalten, müssen dabei weniger oft unterteilt werden als Gebiete mit häufig wechselnder Information.

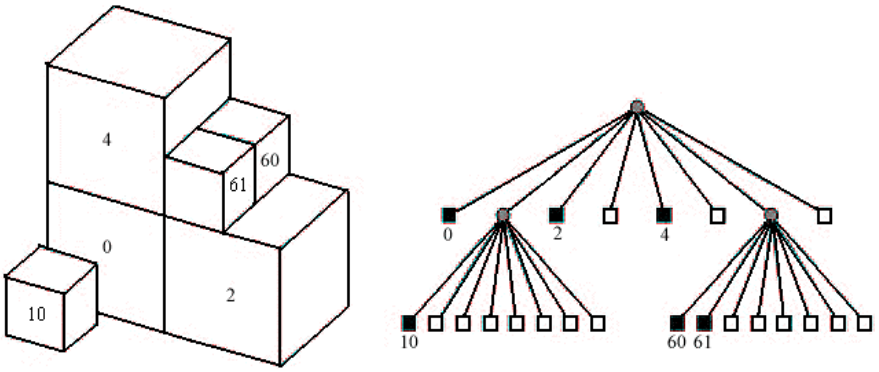

Abbildung 4: Oktalbaum[17]

[16] Vgl. Bungartz, Griebel, Zenger (2002), S.61f.
[17] Vgl. Siller, C. (2002).

Ein Oktalbaum ist letztlich ein aus vielen verschieden großen Würfeln bestehender acht-wertiger Baum. Jeder Teilbaum kann die Information „voll“, „leer“ oder einen weiteren acht-wertigen Teilbaum enthalten. Die „vollen“ Blätter des Baumes können auch weitere Informationen wie Farbe, Transparenz etc. beinhalten.

Ein Oktalbaum benötigt in der Regel deutlich weniger Speicherplatz als ein einfaches Voxel-Modell. Weitere Vorteile sind die einfache Darstellbarkeit und die schnelle räumliche Suche. Allerdings können Objekte mit krummen oder nicht orthogonalen Oberflächen nur mit großem Speicheraufwand approximiert werden. Zudem treten häufig leere Unterbäume und schlecht balancierte Bäume. In letzterem Fall bietet sich die Wahl eines *Binärbaumes* an, bei dem in einem Schritt nur in jeweils einer und nicht in drei Ebenen geteilt wird.

Erweiterte Oktalbäume[18] enthalten zusätzliche Informationen in den Endknoten. Durch die Einführung von *Face Nodes* für Flächen, *Edge Nodes* for Kanten und *Vertex Nodes* für Punkte kann Speicherplatz gespart und das Objekt exakter repräsentiert werden.[19]

Szenegraphen

Szenegraphen sind azyklische Graphen zur Speicherung von Szenenbeschreibungen.[20] Ein Knoten des Szenegraphen enthält einen Teil der Szenenbeschreibung. Knoten *(Shape Nodes, Property Nodes, Group Nodes*) können geometrische Objektbeschreibungen, Materialeigenschaften, Transformationen, Lichtquellenbeschreibungen, Kamerainformationen sowie weitere Attribute (z.B. Farbe, Textur) enthalten. Objekte werden als geordnete Sammlung von Knoten im Szenegraphen gespeichert (vgl. Abbildung 5).

Meistens werden Szenegraphen zu den Volumenmodellen gezählt, jedoch sind prinzipiell als geometrische Objektbeschreibungen auch Punkte, Kanten oder Flächen denkbar. Mittels Gruppierungsknoten werden mehrere Knoten hierarchisch zum Szenegraphen zusammengefasst. Operationen auf Szenegraphen (z.B. Rendering, Auswahl eines Elementes, Suchen, Berechnung von Objektumgebungen) berücksichtigen die durch den Szenegraphen vorgegebene hierarchische Strukturierung der Szenenbeschreibung.

[18] Vgl. Watt (2002), S. 68ff.
[19] Vgl. Gröller, Theußl (2002), S. 18f.
[20] Vgl. Bender, Brill (2003), S. 23f.

Die Repräsentation dreidimensionaler Objekte in Szenegraphen hat große Vorteile für den Rendering-Prozess. Wenn ein Knoten als nicht sichtbar erkannt wird, so können alle Kind-Knoten dieses Knotens aus der Rendering-Pipeline entfernt werden.

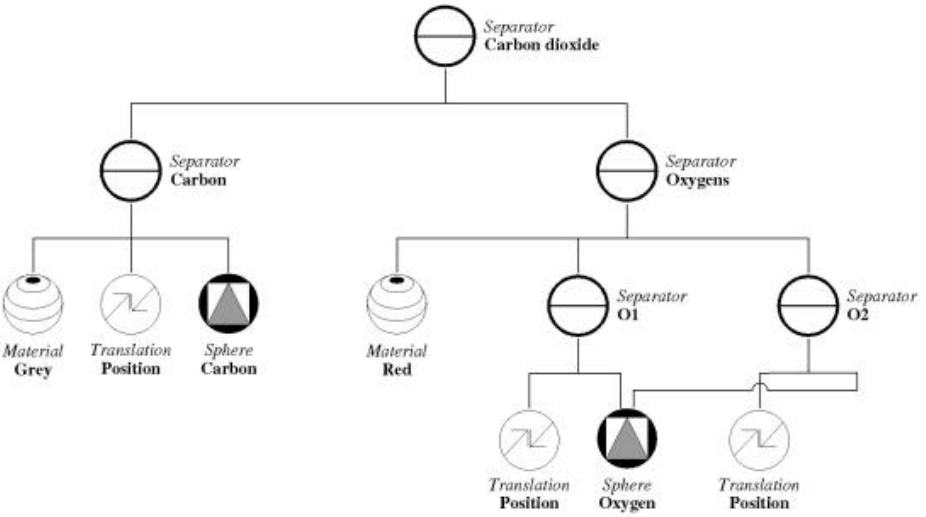

Abbildung 5: Szenegraph eines Kohlendioxid-Moleküls (CO_2)[21]

Constructive Solid Geometry (CSG)

Die Methode des *Constructive Solid Geometry (CSG)* hat sich aus der Erkenntnis ergeben, dass sehr viele Objekte (insbesondere im Fertigungsbereich) durch Kombinationen geometrischer Grundformen dargestellt werden können.[22] Im Unterschied z.B. zum Polygonnetz-Modell ist der CSG-Ansatz eher eine Benutzerdarstellung und erfordert vor der Anzeige spezielle Rendering-Techniken oder die Umformung z.B. in ein Polygonnetz-Modell. Es ist eine Darstellung höherer Ebene, die sowohl zur Erzeugung von Formen dient als auch aufzeichnet, wie sie aufgebaut wurden. Eine Modellgenerierung durch Digitalisierung realer Objekte ist im Allgemeinen nicht möglich.

[21] Vgl. Brickmann, Vollhardt (1995).
[22] Vgl. Watt (2002), S. 62ff.

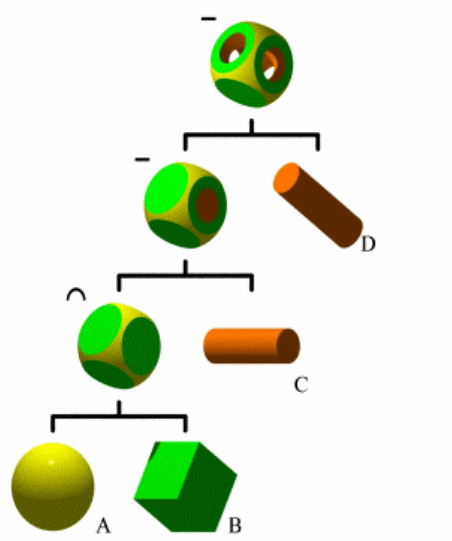

Abbildung 6: Constuctive Solid Geometry[23]

Ein Körper wird als Ergebnis der sukzessiven Booleschen Mengenoperationen Vereinigung, Subtraktion und Schnittmenge und linearen Transformationen auf der Basis einfacher Grundkörper (Primitive) dargestellt. Als Primitive dienen wenige Einheitsobjekte, z. B. Würfel, Kugel, Kegel Zylinder und Torus, jeweils mit den Seitenlängen, Höhen und Radien von 1. Der Konstruktionsprozess wird durch einen Binärbaum mit Primitiven in den Blättern und Operatoren in den inneren Knoten beschrieben (vgl. Abbildung 6). Für die Grundkörper muss gewährleistet sein, dass die erforderlichen Berechnungen (Vereinigung etc.) schnell und effizient realisiert werden können.

Das Konstruktionsprinzip im CSG-Ansatz kann durch eine Grammatik beschrieben werden.[24] Deren Wortschatz entspricht dann genau dem Repräsentationsraum:

```
<Objekt>          ::=   <Primitiv> | <Objekt> <Transformation>
                        | <Objekt> <Operation> <Objekt>
<Transformation>  ::=   Translation | Rotation | Skalierung
<Operation>       ::=   ? | ? |\
<Primitiv>        ::=   Würfel | Zylinder | Kugel | Halbraum ...
```

Die einzigen Informationen, die in den Blättern des Baumes gespeichert werden müssen, sind die Namen der Grundkörper und ihre Abmessungen. Ein Knoten muss den Namen des Operators und die räumlichen Beziehungen zwischen den von ihm kombi-

[23] Vgl. Stewart, N.

nierten Unterknoten speichern. Das CSG-Modell ist die speicherplatzsparendste und exakteste Darstellung.[25]

Die CSG-Methode leidet allerdings auch unter einigen Nachteilen. So ist die benötigte Rechenzeit, um ein gerendertes Bild aus dem Modell zu erstellen, sehr groß. Ein noch größerer Nachteil besteht darin, dass die Methode die Operationen einschränkt, die zur Erzeugung und Veränderung eines Körpers zur Verfügung stehen. Lokale Operationen wie die detaillierte Veränderung einer einzelnen Fläche eines komplexen Objekts können nur schwer durch Mengenoperationen umgesetzt werden.

[24] Vgl. Bungartz, Griebel, Zenger (2002), S. 63.
[25] Vgl. Gröller, Theußl (2002), S. 23.

3 3D-Dateiformate

3.1 Elemente einer Grafik-Datei

Eine Grafik-Datei setzt sich aus einer Sequenz von Dateielementen zusammen. Diese Dateielemente lassen sich in drei Kategorien einordnen: Felder, Tags und Streams.[26]

Ein *Feld* ist ein Dateielement mit fester Größe und Position innerhalb der Grafik-Datei. Die Lage eines Feldes wird entweder durch das Spezifizieren eines absoluten Abstandes von einem bekannten Punkt in der Datei oder durch einen relativen Abstand von einem anderen Dateielement festgelegt. Die Größe eines Feldes ist entweder in der Format-Spezifikation festgelegt oder kann aus anderen Informationen abgeleitet werden.

Im Unterschied zum Feld kann ein *Tag* sowohl in seiner Lage innerhalb der Datei als auch in seiner Größe variieren. Die Lage eines Tags wird wie die eines Feldes festgelegt. Tags können ihrerseits wieder Tags enthalten.

Felder und Tags ermöglichen den wahlfreien Zugriff auf einzelne Dateielemente. Sobald die Position innerhalb einer Datei bekannt ist, kann ein Programm direkt auf das Dateielement zugreifen. Ein Datei, die Daten als *Stream* (Strom) organisiert, muss dagegen sequentiell gelesen werden. Ein Stream setzt sich aus Paketen zusammen, die in ihrer Größe variieren können und eine bestimmte Bedeutung für das die Datei lesende Programm haben. Obwohl Anfang und Ende eines Streams bekannt und spezifiziert sind, ist die Lage der Pakete (mit Ausnahme des ersten Paketes) zumindest vor dem erstmaligen Lesen unbekannt.

Grafik-Dateien, in denen ausschließlich einer der drei genannten Typen vorkommt, sind eher selten. In den meisten Fällen enthält eine Datei zwei oder mehr Typen. Feld-Daten sind in der Regel schneller und einfacher zu lesen als Tag- oder Stream-Daten. Allerdings sind sie weniger flexibel, wenn es darum geht, Daten hinzuzufügen oder zu entfernen. Stream-Daten erfordern weniger Hauptspeicher, um sie zu lesen und zu puffern, als Feld- oder Tag-Daten. Aufgrund des erschwerten Zugriffs ist jedoch schwieriger, bestimmte Daten zu finden.

[26] Vgl. Murray, vanRyper (1996), S. 20ff.

3.2 Klassifizierung der Formate

3D-Formate lassen sich nach ihrem Einsatzzweck und ihren Anforderungen in vier Gruppen unterteilen: CAD/CAM/CAE-Formate, Modellierungs- und Animationsformate, Echtzeitgrafik/VR-Formate und Internet-basierte Formate.

CAD/CAM/CAE-Formate haben die exakte Modellierung von Produkten, Maschinen etc. sowie den einfachen Austausch der Modelle zwischen verschiedenen Programmen zum Ziel. Im Vordergrund steht nicht die realistische Darstellung, sondern die präzise und detaillierte Objektbeschreibung, die auch Grundlage für die Fertigung ist. Die gebräuchlichsten Formate sind DXF (Industrie-Standard), IGES (ANSI-Standard), VDAIS/VDAFS (DIN-Standard), SET (französischer Standard) und STEP (ISO-Standard.

Im Unterschied zu den CAD/CAM/CAE-Formaten werden bei den *Modellierungs- und Animationsformaten* nicht nur einzelne Objekte, sondern auf der Grundlage einer Szenenbeschreibungssprache (*Scene Description Language, SDL*) komplette Szenen mit Lichtern, Kamera, Effekten etc. beschrieben. Oft bestehen spezielle Ausdrucksmöglichkeiten für die Modellierung natürlicher Phänomene wie Fell, Feuer, Rauch, Kleidung etc. Ziel sind meist gerenderte Bilder oder Computeranimationen, was eine geringere Präzision als die der CAD/CAM/CAE-Formate akzeptabel macht. Modellierungs- und Animationsformate sind meist Austauschformate, weisen aber zunehmend auch prozedurale Bestandteile auf. Beispiele für diese Klasse von 3D-Formaten sind Alias Wavefront Object (OBJ), 3D Studio (3DS) und Persistance of Vision (POV).

Zu der Szenenmodellierung der Modellierungs- und Animationsformate treten bei den *Echtzeitgrafik/VR-Formaten* komplexere Animationen, Simulationen und Interaktionen sowie die Unterstützung spezieller Ein- und Ausgabegeräte. Zudem bietet die Virtuelle Realität nicht wie die Animation eine passive Betrachtung von 3D-Formaten, sondern eine blickpunktunabhängige Echtzeitberechnung von 3D-Welten, die es Nutzern ermöglicht, durch eine computergenerierte räumliche Umgebung zu navigieren und mit dieser zu interagieren. Auflösung und Detaillierungsgrad sind verhältnismäßig gering. Szenenbeschreibungs-Formate im Echtzeitgrafik-Sektor sind oft Szenegraph-basiert (vgl. Kapitel 2.4). Wichtige Echtzeitgrafik/VR-Formate sind OpenGL, Direct3D und QuickDraw3D.

Internet-basierte 3D-Grafik-Formate ermöglichen die plattformunabhängige Darstellung dreidimensionaler Welten durch Browser-PlugIns oder Applets. Aufgrund der oft geringen Bandbreite zwischen Client und Server sind Ansätze der Modularisierung, der Kompression, des Streaming und der Skalierung notwendig. Bei den Internet-basierten 3D-Grafik-Formaten haben sich bisher kaum Standards durchgesetzt. Wichtige Vertreter dieser Gruppe sind VRML, X3D und MPEG-4.

Im Folgenden werden einige der derzeit gängigsten 3D-Formate, namentlich DXF, OBJ, 3DS, VRML und X3D, vorgestellt und den beschriebenen Gruppen zugeordnet. Mit Ausnahme von 3DS sind die Spezifikationen aller Formate im Internet erhältlich.

3.3 Autodesk Drawing Interchange Format (DXF)

Das Dateiformat DXF ist ein programmspezifisches Format der Firma Autodesk. Es wird in versionsabhängigen Formaten von dem Programm AutoCAD erzeugt. Damit gehört es zur Gruppe der CAD/CAM/CAE-Formate. Mit dem weltweiten Erfolg von AutoCAD ist DXF zum De-facto-Standard in der Industrie geworden und wird besonders häufig zum Austausch zwei- und dreidimensionaler Drahtmodell-Daten genutzt. Zudem verwendet DXF häufig Polygone, bikubische Parameteroberflächen und CSG-Repräsentationen. Eine offizielle Dokumentation ist im Internet verfügbar.[27]

Eine DXF-Datei ist eine komplette Repräsentation der AutoCAD-Zeichnungs-Datenbank, weshalb einige Features bzw. Konzepte nicht von anderen CAD-Systemen genutzt werden können. Jede Datei besteht aus vier Sektionen: Kopfteil (*Header*), Tabellen (*Table*), Blockdefinitionen (*Block*) und geometrischer Inhalt der Zeichnung (*Entity*).

Der Kopfteil enthält allgemeine Informationen über die Zeichnung und dient der Definition einer spezifischen Arbeitsumgebung. Hier werden Variablennamen und -werte festgelegt. Die Tabellen definieren Linientypen, Layer, Textstil, Koordinatensysteme etc. Im Abschnitt für Blockdefinitionen erlaubt es DXF, geometrische Grundobjekte in Blöcken zu komplexen Objekten zusammenzufassen. Ein solcher Block kann später beliebig oft als Einzelobjekt in der DXF-Datei referenziert werden. Die Entity-Sektion

[27] Vgl. o.V.[1]

enthält schließlich die geometrischen Daten der Zeichnung. Hier werden die Kanten, Polygone, bikubische Parameteroberflächen usw. mit verschiedenen Attributen (Layerzugehörigkeit, Farbe, Strichelung) gespeichert.

3.4 Wavefront Object (OBJ)

Wavefront Object-Dateien werden von Wavefront's Advanced Visualizer-Anwendung zur Speicherung dreidimensionaler geometrischer Objekte genutzt.[28] Im Advanced Visualizer werden geometrische Objektdateien im ASCII-Format (.obj) oder im Binärformat (.mod) gespeichert. Beide Formate gehören zu den Modellierungs- und Animationsformaten. Da das Binärformat proprietär und nicht dokumentiert ist, wird hier nur das ASCII-Format beschrieben.[29]

Das OBJ-Dateiformat unterstützt Linien, Polygone, Freiform-Kurven und Oberflächen. Linien und Polygone werden mit Hilfe ihrer Punkte beschrieben, während Kurven und andere Oberflächen durch Kontrollpunkte und andere vom Typ der entsprechenden Kurve abhängigen Informationen beschrieben werden.

OBJ-Dateien erfordern keinen Header, jedoch ist es üblich, die Datei mit einer Kommentarzeile (#) zu beginnen. Jede Zeile beginnt mit einem Schlüsselwort, welches von Daten für dieses Schlüsselwort gefolgt werden kann. Zeilen werden bis zum Ende der Datei gelesen und verarbeitet.

Die meisten OBJ-Dateien enthalten ausschließlich polygonale Daten. Um ein Polygon zu beschreiben, wird zunächst jeder Eckpunkt des Polygons mit dem Schlüsselwort `v` und anschließend die sich aus den Eckpunkten ergebende Fläche mit dem Schlüsselwort `f` definiert. Die Zeile des f-Befehls enthält eine Aufzählung der Eckpunkte des Polygons in der Reihenfolge, in der sie in der Datei auftreten. Das folgende Beispiel beschreibt ein einfaches Dreieck:

[28] Vgl. Murray, vanRyper (1996), S. 946.
[29] Vgl. o.V.[3].

```
# Simple Wavefront file
v  0.0  0.0  0.0
v  0.0  1.0  0.0
v  0.0  0.0  1.0
f  1  2  3
```

OBJ-Dateien enthalten keine Farbdefinitionen für Flächen, jedoch können Flächen Materialien referenzieren, die in einer separaten Material-Bibliothek-Datei (*material library file*) gespeichert sind. Die Material-Bibliothek kann mit Hilfe des Schlüsselworts `mtllib` geladen werden. Die Material-Bibliothek enthält die Definitionen für die RGB-Werte für die Farben des Materials und andere Charakteristika wie Lichtbrechung, Transparenz etc.

Die OBJ-Datei referenziert Materialien mit Hilfe des `usemtllib`-Schlüsselwortes. Allen folgenden Flächen werden die Attribute dieses Materials zugeordnet, bis der nächste `usemtl`-Befehl erfolgt.

Flächen können mit Hilfe des Schlüsselwortes `g` bestimmten Gruppen zugeordnet werden. Damit können leicht zu handhabende Sub-Objekte geschaffen werden, um das Ändern und Animieren von 3D-Modellen zu erleichtern. Flächen können auch zu mehr als einer Gruppe gehören.

3.5 Autodesk 3D Studio (3DS)

Ebenfalls zur Gruppe der Modellierungs- und Animationsformate gehören die Formate von Autodesk 3D Studio, einem weit verbreiteten High-End-Grafikprogramm für Modellierung und Rendering von 3D-Szenen. Da Autodesk anders als im Fall von AutoCAD keine Informationen über die Formate von 3D Studio offen legt, beziehen sich die folgenden Ausführungen nicht auf eine Spezifikation des Herstellers und können keinen Anspruch auf Richtigkeit erheben.

Es gibt zwei Datenformate, die von 3D Studio genutzt werden: Ein Binärformat mit der Dateierweiterung .3ds und ein ASCII-Format, welches die Dateierweiterung .asc trägt. Beide Formate gehören zu den Modellierung- und Animationsformaten werden in der Praxis als Austauschformate genutzt. Hier soll nur das Binärformat beschrieben werden.

3DS-Dateien bestehen aus Tags (vgl. Kapitel 3.1), die von Autodesk *Chunks* (engl. für Brocken, Klotz) genannt werden.[30] Rendering-Anwendungen oder Dateiumwandlungs-Programme sind in der Lage, aus den Chunk-Typen die Erscheinung und das Verhalten der in der Datei gefundenen Objekte abzuleiten. Jeder Chunk beginnt mit einem ID-Wert (`Chunk_ID`), der die Art der in diesem Chunk enthaltenen Daten festlegt. Darauf folgt die Länge des Chunks (`Chunk_length`). Diese legt fest, inwieweit die folgenden Daten diesem Chunk zugeordnet werden. Die diesem Chunk zugeordneten Daten können dabei mehrere Chunks umfassen. Die eigentlichen Daten des Chunks folgen dem ID-Wert und der Abstandsinformation.

```
Typedef struct _CHUNK
{
WORD    Chunk_ID;       /* Chunk-Typ */
DWORD   Chunk_length;   /* Rel. Abstand in Bytes zum nächsten Chunk */
} CHUNK;
```

Chunks lassen sich in Primär-Chunks (`Primary`), Haupt-Chunks (`Main`) und untergeordnete Chunks (`Subordinate`) klassifizieren und sind folgendermaßen in einer Hierarchie angeordnet:[31]

```
Primär-Chunk
      Haupt-Chunk 1
            Untergeordneter Chunk 1
            Untergeordneter Chunk 2
            ...
      Haupt-Chunk 2
            Untergeordneter Chunk 1
            Untergeordneter Chunk 2
            ...
      Haupt-Chunk 3
```

Die Haupt-Chunks sind im Primär-Chunk und die untergeordneten Chunks in jeweils einem Haupt-Chunk enthalten. Jede Datei enthält nur einen Primär-Chunk, der sich am Beginn der Datei befindet. Darauf folgt der erste Haupt-Tag. Das `Chunk_length`-Feld

[30] Vgl. Bourke, P. (1996)
[31] Vgl. Murray, vanRyper (1996), S. 283.

des Haupt-Chunk-Tags ist im Allgemeinen der Abstand zum nächsten Haupt-Chunk. Eine Liste bekannter Chunk-Typen findet sich in MURRAY, VANRYPER (1996), S. 284ff.

3.6 Virtual Reality Modeling Language (VRML)

Die Virtual Reality Modeling Language (VRML) ist ein Internet-basiertes 3D-Grafik-Format, das es erlaubt, dreidimensionale Szenen darzustellen.[32] Entwickelt wurde VRML Mitte der Neunziger Jahre vom Web3D-Konsortium (damals noch VRML Architecture Group(VAG)).[33] Für die Darstellung einer Szene benötigt man ein VRML-PlugIn für einen Internet-fähigen Browser.

Die Grundlage zur Beschreibung einer VRML-Datei ist der Szenegraph (vgl. Kapitel 2.4). Die Knoten Szenegraphen beschreiben die Objekte und Eigenschaften der 3D-Welt wie geometrische Körper, Animationen oder Lichtquellen. Durch Gruppenknoten lassen sich einzelne Knoten zusammenfassen.

Am Anfang jeder VRML-Datei steht der VRML-Header, der Versionsnummer und verwendeten Zeichensatz festlegt. Es folgt ein einfaches Codebeispiel.

```
#VRML V2.0 utf8

Transform {
  children [
    Shape {
      appearance Appearance {
        material Material {
          diffuseColor 0.0 0.0 0.9
        }
      }
      geometry Box {size 6.0 6.0 3.0}
    }
  ]
}
Transform {
  translation 10.0 0.0 0.0
    children [
      Shape {
        appearance Appearance {
          material Material {
            diffuseColor 0.6 0.1 0.1
          }
        }
        geometry Sphere {radius 4}
      }
```

[32] Zur ausführlichen Behandlung von VRML vgl. z.B. Däßler, Palm (1998).
[33] www.web3d.org. Die Spezifikation findet sich unter o.V.[2].

```
    ]
}
```

Knoten enthalten einen Knotentyp, gefolgt von beliebig vielen Feldern. Die Felder definieren die Knotenattribute; sie beschreiben die statischen Objekteigenschaften - hier z.B. die Seitenlängen eines Würfels oder den Radius einer Kugel. Der wichtigste Knotentyp ist der `Shape`-Knoten. Durch ihn werden Objekte und deren Aussehen definiert. Shape-Objekte werden standardmäßig immer in den Ursprung des Koordinatensystems gesetzt. Mit Hilfe des Transformationsknotens (`transform`) kann man alle ihm untergeordneten Objekte verschieben (`translation`), drehen (`rotation`) und skalieren (`scale`).

VRML-Daten werden stets als ASCII-Files gespeichert. Dadurch lassen sich die Dateien zwar sehr leicht erzeugen und manipulieren, brauchen jedoch viel Speicherplatz. Die Dateien wachsen mit der Detailliertheit, Komplexität und Anzahl der modellierten Objekte schnell zu nur noch schwer handhabbaren Datenmengen an. Nachteilig ist auch, dass die Dateien erst nach komplettem Download sichtbar sind. Es kann vorkommen, dass verschiedene Browser-PlugIns ein und dieselbe VRML-Datei unterschiedlich darstellen.

3.7 eXtensible 3D (X3D)

Um die Nachteile von VRML auszuräumen, hat das Web3D-Konsortium XML zur Formulierung einer Beschreibungssprache verwendet und im August 2001 die DTD eXtensible 3D (X3D) vorgestellt.[34] Durch die Unterstützung von XML kann bei 3D-Szenen der Inhalt der Szene von der Darstellung getrennt und somit mehr Flexibilität erreicht werden.

Prinzipien wie die Beschreibung einer Szene als hierarchischer Szenegraph oder die Methoden der Animation und Interaktion bleiben aus VRML erhalten. Im Unterschied zu VRML ist X3D jedoch modular aufgebaut (vgl. Abbildung 7).

[34] Vgl. o.V.[4]

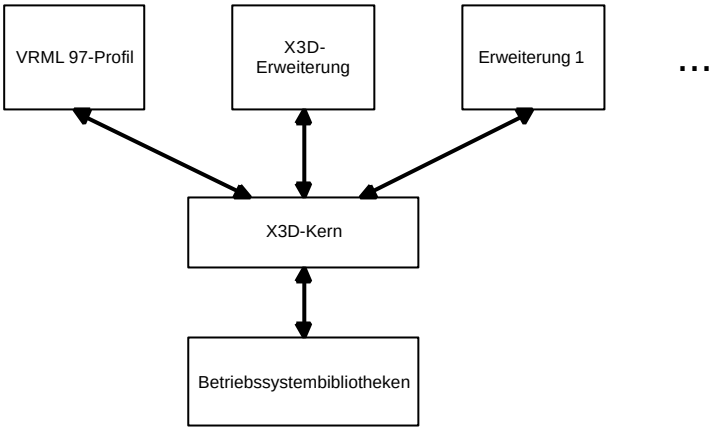

Abbildung 7: Modularer X3D-Aufbau

Der *X3D-Kern* ähnelt VRML, enthält jedoch nur die notwendigsten Spezifikationen. Einige Eigenschaften wurden entfernt und andere hinzugefügt, um X3D leistungsfähiger zu machen.

Im *VRML 97-Profil* sind alle Eigenschaften von VRML spezifiziert, so dass zusammen mit der Kernspezifikation die volle VRML-Funktionalität gewährleistet wird. Damit ist X3D voll abwärtskompatibel und es existieren einige Konverter, die VRML in X3D und umgekehrt umwandeln.

Die *X3D-Erweiterung* enthält einige Neuerungen wie NURBS (*Non Uniformal Rational B-Splines*) und Streaming.

Der große Vorteil von X3D ist, das es sich nahezu unbegrenzt erweitern lässt. Zu den Funktionen des 3D-Kerns werden weitere Profile angelegt, die auf die speziellen Bedürfnisse der Anwender eingehen, ohne dass die Erweiterungen wie bei VRML mit proprietären Programmen hinzugefügt werden.

Als Codebeispiel dient hier das VRML-Beispiel aus Kapitel 3.6, um die Ähnlichkeit der beiden Formate herauszustellen. Der Syntaxbaum stimmt ebenso wie die Mehrzahl der Knoten überein. In der ersten Zeile werden wieder Versionsnummer und der verwendete Zeichensatz angegeben. Statt der Knotennamen in Verbindung mit geschweiften Klammern verwendet X3D XML-Start-Tags und -End-Tags.

```
<?xml version="1.0" encoding="UTF-8"?>

<X3D>
  <Scene>
    <Transform>
      <Shape>
        <Appearance>
          <Material diffuseColor="0.0 0.0 0.9"/>
        </Appearance>
        <Box size="8.0 6.0 3.0"/>
      </Shape>
    </Transform>
    <Transform translation="10.0 0.0 0.0">
      <Shape>
        <Appearance>
          <Material diffuseColor="0.6 0.1 0.1"/>
        </Appearance>
        <Sphere radius="4.0"/>
      </Shape>
    </Transform>
  </Scene>
</X3D>
```

X3D soll alle 3D-Standards auf einen Nenner bringen. Der Standard ist offen, plattformunabhängig und lizenzgebührenfrei. Spezielle Features können als Pakete eingebunden werden und müssen nicht von proprietären Programmen realisiert werden. Allerdings sind X3D-Dateien noch etwas größer als VRML-Dateien. Zudem ist der Standard noch relativ jung und wird erst durch wenige und z.T. nicht ausgereifte Programme genutzt.

4 Fazit

Diese Ausarbeitung befasste sich mit der computergestützten Repräsentation dreidimensionaler Objekte und Szenen. Dazu wurden im zweiten Kapitel Grundlagen der dreidimensionalen Objekt- und Szenenrepräsentation behandelt und im dritten Kapitel eine Einteilung der 3D-Formate in vier Gruppen vorgenommen. Anschließend wurden fünf der gängigsten Formate vorgestellt.

Die Vielzahl der Bereiche, in denen dreidimensionale Computergrafik heute eingesetzt wird, zeigt, dass der große Forschungsaufwand berechtigt ist. Die Anwendungsmöglichkeiten reichen von Computerspielen und dreidimensionale Chatrooms über virtuelles Lernen, Produktvisualisierung, CAD und Industriedesign, der Besichtigung virtueller Gebäude oder Städte bis hin zu methodischen Bereichen wie Medizin, Geologie und Stadtplanung. Insbesondere die wissenschaftlichen Disziplinen finden bei ihrer Suche nach naturnaher und realistischer Verwaltung und Analyse oft in der dreidimensionalen Computergrafik eine Antwort. Hier spielt der Ansatz der Volumengrafik eine immer größere Rolle.

Bezüglich der dreidimensionalen Dateiformate lässt sich sagen, dass sie stets unter Berücksichtigung ihres Einsatzzweckes untersucht werden müssen; ein Vergleich zwischen Formaten verschiedener Gruppen, z.B. zwischen einem CAD/CAM/CAE-Format und einem Internet-basierten Format, ist in der Regel nicht sinnvoll. Die Entscheidung für das eine oder andere Format muss dann immer mit Blick auf die Zielgruppe erfolgen; die technischen Möglichkeiten sind nur zweitrangig.

Auch muss festgestellt werden, dass eine Zuordnung von Grafikformaten zu einer 3D-Klasse, aber auch allgemein zu den 3D-Formaten häufig nicht leicht fällt. Ein Grund hierfür ist, dass 3D-Daten heute von einer Reihe von Formaten unterstützt werden, die in der Vergangenheit nur die Speicherung zweidimensionaler Daten ermöglichten.

Zu erkennen ist, dass sich bisher kaum einheitliche Standards durchgesetzt haben. In einem Großteil aller Fälle, in denen ein neues Grafikformat entwickelt wird, treten keine neuen Funktionalitäten hinzu, so dass hier die Weiterverwendung eines vorhandenen Formats völlig ausreichend gewesen wäre. In der Literatur wird vor allem X3D aufgrund der bekannten Vorteile von XML eine hohe Verbreitung vorausgesagt.

Literaturverzeichnis

Bender, M.; Brill, M.: *Computergrafik. Ein anwendungsorientiertes Lehrbuch*. München 2003.

Bourke, P.: *3D-Studio File Format (.3ds)*. 1996. http://astronomy.swin.edu.au/~pbourke/geomformats/3ds/. Abrufdatum 2003-10-26.

Brickmann, J.; Vollhardt, H.: *3D Molecular Graphics on the World Wide Web*. 1995. http://ws05.pc.chemie.tu-darmstadt.de/research/vrml/psb95/. Abrufdatum 2003-12-29.

Bungartz, H.-J.; Griebel, M.; Zenger, C.: *Einführung in die Computergraphik. Grundlagen, Geometrische Modellierung, Algorithmen*. Braunschweig 1996.

Däßler, R.; Palm, H.: *Virtuelle Informationsräume mit VRML. Informationen recherchieren und präsentieren in 3D.* Heidelberg 1998.

Gilchrist, T.: *Creating Spline Curves*. 2001. http://www.tonyg.info/tutorials/LW6/Model/Splines_and_Stuff.htm. Abrufdatum 2003-28-11.

Gröller, E.; Purgathofer, W.: *Grafische Datenverarbeitung*. In: Informatik-Handbuch. Hrsg.: Rechenberg, P.; G. Pomberger. München 1999.

Gröller, E.; Theußl, T.: *3D-Datenstrukturen*. 2002. http://www.cg.tuwien.ac.at/courses/CG2/ SS2002/AdvancedDataStructures.pdf. Abrufdatum 2003-12-15.

Häuser, S.: *Generierung, Darstellung und Interaktion mit Voxel*. 1998. http://elib.uni-stuttgart.de/opus/volltexte/1999/354/pdf/354_1.pdf. Abrufdatum 2003-12-15.

Hughes, D.: *COSC 4F90 Project: Voxel Sculptor*. 1997. http://www.cosc.brocku.ca/Project/ info/voxel.htm. Abrufdatum 2003-11-28.

Murray, J.D.; vanRyper, W.: *Encyclopedia of Graphics File Formats, Second Edition*. Sebastopol 1996.

o.V.[1]: *AutoCAD 2000 DXF Reference*. http://www.autodesk.com/techpubs/autocad/acad2000/dxf/index.htm. Abrufdatum 2003-12-02.

o.V.[2]: *VRML97 International Standard Specification*. http://www.web3d.org/technicalinfo/ specifications/ISO_IEC_14772-All/index.html. Abrufdatum 2003-12-02.

o.V.[3]: *Wavefront Object Specification*. 1996. http://netghost.narod.ru/gff/vendspec/waveobj/index.htm. Abrufdatum 2003-12-02.

o.V.[4]: *X3D Specification*. http://www.web3d.org/specifications/ISO-IEC-19775/ index.html. Abrufdatum 2003-12-04.

Pfund, M.: *Geometrische Modellierung dreidimensionaler Objekte in Geo-Informationssystemen*. Zürich 1999.

Siller, C.: *Octree-Generierung anhand rotierender Objekte.* 2002. http://www.informatik.uni- maheim.de/informatik/pi4/stud/veranstaltungen/ws200203/seminar_data/Cornelius Siller.pdf. Abrufdatum 2003-12-02.

Stewart, N.*: Hardware Based CSG Rendering*. http://goanna.cs.rmit.edu.au/ ~nigels/. Abrufdatum 2003-12-02.

Watt, A.: *3D-Computergrafik*. München 2002.