Projektpartner: Westfälische Wilhelms-Universität Münster - Institut für Geoinformatik
Datum: 03.07.2003

Lerneinheit 2: „Grundlagen der Computergrafik"

Einleitung

Ziel der zweiten Lerneinheit ist es, dem Studierenden die informatorischen Grundlagen der Computer-Grafik zu vermitteln. Es werden dabei nur die Teilgebiete der Computer-Grafik angesprochen, die für die 3D-Geovisualisierung von zentraler Relevanz sind. Neben den grundlegenden Methoden zur Beschreibung von 3D-Welten sind dies insbesondere die verschiedenen verfügbaren Verfahren zur Bildgenerierung. Zudem sollen technische Termini eingeführt werden, deren Kenntnis für die praktische Anwendung der 3D-Geovisualisierung sinnvoll ist.

bmb+f Bundesministerium für Bildung und Forschung
Das Projekt Geoinformation - Neue Medien für die Einführung eines neuen Querschnittsfaches ist ein Projekt im Rahmen des BMBF-Programms Neue Medien in der Bildung.

zip Ein Projekt im Zukunftsinvestitionsprogramm der Bundesregierung
Das Projekt wird im Rahmen des Zukunftsinvestitionsprogramms der Bundesregierung (zip) gefördert.

Inhalt

Grundaufgabe der Computer-Grafik

Die Computer-Grafik befasst sich mit der Computer-gestützten Generierung bildhafter Darstellungen aus (alpha-) numerischen Daten für im Prinzip beliebige Objekte (z.B. Gebäude, Maschinen-Bauteile, statistische Daten, Landschaften etc.). Computer-grafische Methoden werden in zahlreichen Bereichen eingesetzt, u.a. in Architektur, Design, Medizin, Maschinenbau, Wirtschaftswesen sowie in den verschiedenen Geowissenschaften und der Planung.

Abbildung 1: Computergrafik-Beispiele

Die Grundaufgabe der Computer-Grafik besteht in der Transformation Computer-basierter Repräsentationen realer oder imaginärer Objekte in eine bildhafte Darstellung. Im Vordergrund des Interesses steht somit die Bild-"Synthese" (im Gegensatz zur Bild-Analyse, welche sich z. B. mit der Merkmalsextraktion aus gescannten Bildern beschäftigt).

Die grafische Darstellung muss dabei so konzipiert werden, dass die hinter den (alpha-) numerischen Daten stehende Information in der Grafik spontan wahrgenommen und im gewünschten (richtigen) Sinne verstanden wird (visuelle Perzeption, Erfassung des relevanten Sachverhaltes mit einem Blick). Diese Aufgabe fällt allerdings weniger in den Bereich der Computer-Grafik als vielmehr in den Bereich des "Visualisierungsentwurfs".

Grafik-Bibliotheken

Die Programmierung Computer-grafischer Funktionen ist sehr aufwändig. Daher werden in der Praxis in der Regel bestehende Grafik-Pakete genutzt. Derartige Grafik-Pakete bieten Funktionen zum Zeichnen geometrischer Primitive wie Punkten, Linien, Polygonen, Oberflächen, Texten, Bitmaps usw., Funktionen zur Handhabung verschiedener Farbmodelle, Festlegen des Betrachterstandortes ("Kamera-Position") u. v. m. Bekannte Grafik-Pakete sind z. B. PHIGS (Programmer's Hierarchical Interactive Graphics System), GKS (Graphic Kernel System) oder die OpenGL (Open Graphics Language). Die OpenGL stellt z. Zt. praktisch den De-facto-Industriestandard für 3D-Anwendungen dar. Zunehmende Verbreitung findet heute auch die Direct3D-Bibliothek.

Eine wesentliche Fähigkeit aktueller 3D-Grafikpakete besteht in der Möglichkeit der Erzeugung von Darstellungen, die einen dreidimensionalen, "realitätsgetreuen" Raumeindruck vermitteln. Der Vorgang der Erzeugung derartiger plastischer Darstellungen von Objekten wird häufig als Rendering bezeichnet (engl. to render = übersetzen). Die erreichbare Bildqualität hängt dabei von der Art der jeweils eingesetzten Schattierungsverfahren ab (s. Kap. „Ablauf des Rendering").

Die Programmierung mit Hilfe der genannten Bibliotheken ist recht mühsam. Daher werden statt "Low-level"-Grafik-Bibliotheken häufig Werkzeuge eingesetzt, die auf einem höheren Abstraktionsniveau operieren. Hierunter fallen zum Beispiel die Szenengraph-basierte Bibliothek Java 3D, die Bibliothek vtk oder die Entwicklungsumgebung AVS/Express. Diese Werkzeuge nutzen allerdings im Regelfall Low-level-Bibliotheken wie OpenGL oder Direct3D.

Geometrische Primitive

Die Objekte, in denen die seitens des Renderers benötigte geometrische Information spezifiziert wird, sollen im Weiteren als Shapes bezeichnet werden. In vielen Entwicklungsumgebungen lassen sich die beiden folgenden Shape-Kategorien unterscheiden:

1. Vertex-basierte Shapes:
Gängig ist beispielsweise eine Unterteilung in Punkte/Punktmengen, Linien/Linienmengen und Dreiecke sowie teilweise Vierecke als grundlegende Typen. Darüber hinaus werden Arrays für Punkte, Linien, Drei-/Vierecke und Strip-Arrays verwendet. Strip-Arrays wie "line strips", "triangle strips", "triangle fans" etc. erlauben die effiziente Spezifikation zusammenhängender Dreiecke, indem gemeinsame Vertizes und/oder Kanten implizit festgelegt werden. Als weiterer Typ sind indizierte Geometrien zu nennen, in welchen Arrays für die Vertex-Information und die zugehörige Vermaschung gehalten werden, wodurch sich die unnötige doppelte Speicherung von Vertizes vermeiden lässt.

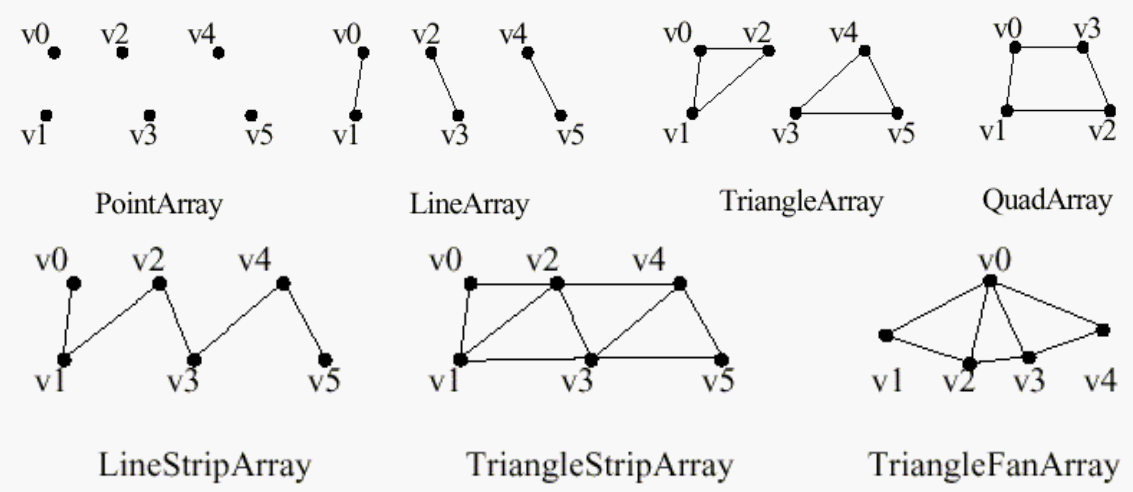

Abbildung 2: Vertex-Shapes

Vertex-basierte Shapes sind durch ihre Vertizes (Stützpunkte) und mit diesen assoziierten Daten wie Normalenvektoren (Richtungsvektoren), Farbwerten und Texturinformation charakterisiert. Die in Geovisualisierungen häufig darzustellenden Gitternetze und TINs lassen sich mit Hilfe dieser Strukturen zusammensetzen. Der häufige Fall, dass sich eine Farbangabe nicht auf einen Vertex, sondern auf eine durch Kanten gebildete Fläche bezieht, ist ebenfalls umsetzbar.

2. Analytische Shapes:
Objekte dieses Typs werden durch ihre analytischen Parameter definiert, zum Beispiel eine Kugel durch ihren Mittelpunkt und Radius. Für häufig benötigte geometrische Objekte wie Quader, Kugeln, Zylinder, Gitternetze etc. lassen sich Klassen bereitstellen, in welchen die Komposition aus den verfügbaren Grundobjekten gekapselt ist. Einige Renderer können verschiedene analytische Beschreibungen direkt verarbeiten, zumeist ist jedoch eine Umsetzung in einfache Grundobjekte wie Dreiecke erforderlich.

Kamera-Modell

Innerhalb des dreidimensionalen Darstellungsraums kann der Anwender die Betrachterposition (den Ansichtspunkt), die Blickrichtung und das "Blickfeld" (Höhenwinkel und Breitenwinkel) frei wählen. Zumeist wird hierzu ein vereinfachtes Modell der "synthetischen Kamera" verwendet, in dem der sichtbare Raumausschnitt die Form eines vierseitigen Pyramidenstumpfes hat.

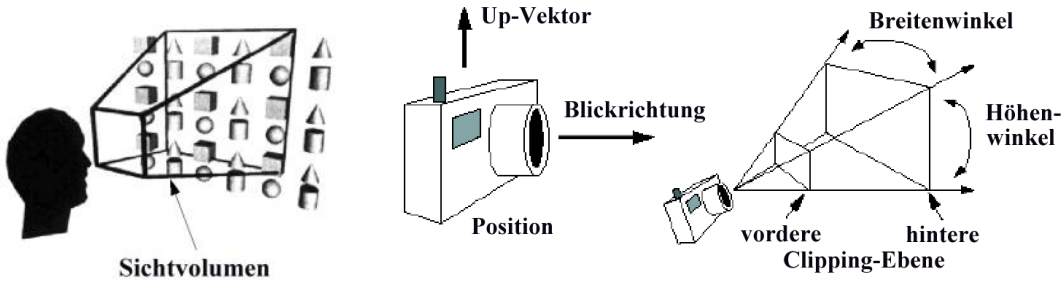

Abbildung 3: Kamera-Modell

Für wachsende Höhen- und Breitenwinkel nimmt die perspektivische Verzerrung der Darstellung zu (insofern keine Parallelprojektion erfolgt). Festzuhalten ist, dass dem generierten Bild bei perspektivischer Projektion kein einheitlicher Maßstab zugrunde liegt, das heißt, ein fester Faktor beispielsweise für die Umrechnung von Bildschirmkoordinaten in Realwelt-Entfernungen lässt sich nicht angeben.

Für die Bildgenerierung ist die Definition einer Kamera erforderlich. In den Kameraobjekten sind die oben genannten Parameter (Ansichtspunkt, Blickrichtung, Blickfeld) zusammengefasst. Zur Änderung dieser Parameter stehen Methoden zur Verfügung, welche die Umsetzung intuitiv bedienbarer Werkzeuge zur Navigation durch den Darstellungsraum erlauben.

Ablauf des Rendering

Die Arbeitsschritte des Rendering-Prozesses sollen an Hand eines einfachen Beispiels erläutert werden. Dargestellt werden soll eine einfache 3D-Welt, die aus einer rechteckigen Bodenfläche, einem darauf positionierten 3D-Gebäude und zwei Straßenzügen besteht.

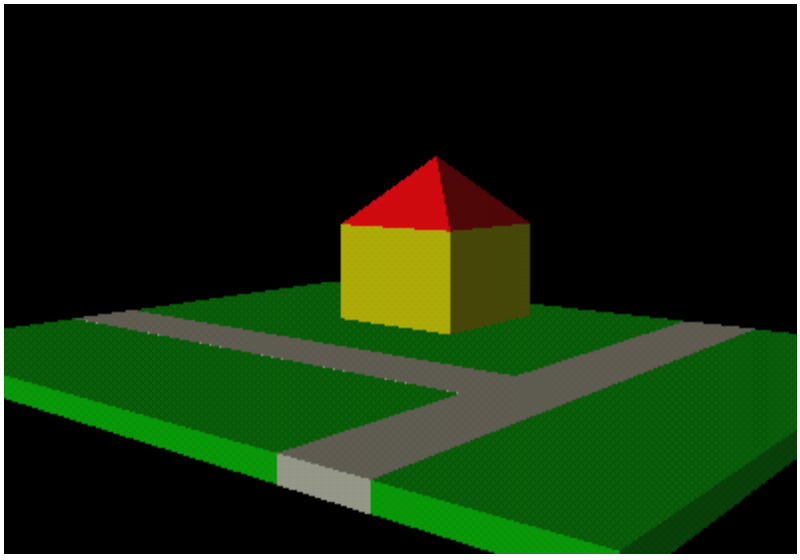

Abbildung 4: Beispielhafte 3D-Welt

Im Weiteren sollen die einzelnen Arbeitsschritte erläutert werden, die zum plastisch wirkenden 3D-Bild führen.

Schritt 1: Modellbeschreibung

Zunächst ist die 3D-Welt geometrisch zu beschreiben:

- Gebäude: Im vorliegenden Fall liegt der Gebäudegrundriss als Rechteck vor, dessen Eckpunkte P1, P2, P3 und P4 in einem Georeferenzsystem gegeben sind. Zusätzlich sind für das Gebäude die Traufenhöhe h1 sowie die Höhe h2 des Zeltdachs anzugeben.

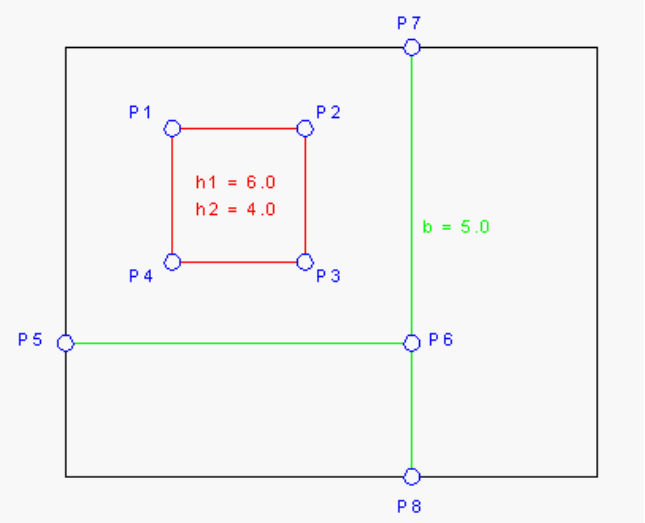

Abbildung 5: Geometrische Beschreibung der 3D-Welt

- Straßen: Die beiden Straßen-Elemente liegen jeweils als Linien vor, die durch ihren Start- und Endpunkt beschrieben werden ("Straßenachsen", P5 und P6 bzw.P7 und P8). Zudem ist die Breite b der Straße anzugeben.
- Bodenfläche: Die rechteckige Bodenfläche ist durch die Angabe der vier Eckpunkt-Koordinaten angegeben.

Zu beachten ist, dass die Positions- und Abstandsangaben zueinander kompatibel sein müssen. Beispielsweise wären Positionsangaben in geografischer Länge und Breite und Abstandsangaben in Metern nicht ohne weiteres innerhalb des Koordinatensystems darstellbar. Positionsangaben in Form von Gauß-Krüger-Koordinaten in Metern hingegen könnten direkt verarbeitet werden.

Neben der geometrischen Beschreibung sind auch die visuellen Ausprägungen der Geometrien festzulegen:

- Gebäude: Das Gebäude soll als dunkelgelber Quader mit rotem Dach dargestellt werden.
- Straßen: Die Straßen-Elemente sind in grauer Farbe und flächig gefüllt darzustellen.
- Bodenfläche: Für die Bodenfläche wird eine dunkelgrüne Farbe spezifiziert.

Schritt 2: Zerlegung der Geometrien in Primitive

Die meisten Rendering-Systeme gehen so vor, dass die Geometrien der darzustellenden Elemente in einfach zu berechnende Primitive, zum Beispiel in Dreiecke oder andere einfache Polygone, zerlegt werden, die im Weiteren einzeln anstatt der komplexen Geometrien berechnet werden. Abbildung 6 zeigt die Zerlegung der spezifizierten 3D-Welt in Dreiecksprimitive.

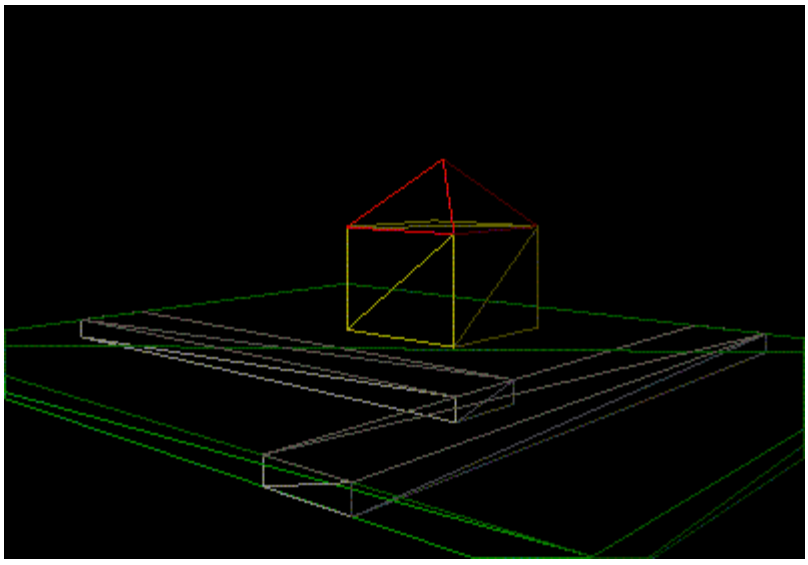

Abbildung 6: Zerlegung in Dreiecksprimitive

Schritt 3: Rasterisierung

Für den aktuellen Ansichtspunkt wird ein zur Projektionsfläche gehöriges, regelmäßiges Raster generiert. Aus Gründen der besseren Erkennbarkeit wird die Vorgehensweise an Hand eines groben Rasters (49 x 34 Rasterzellen) demonstriert.

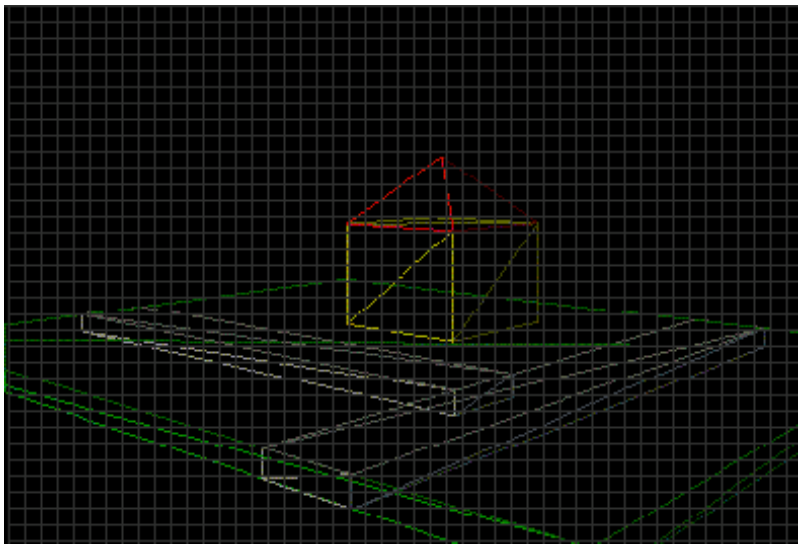

Abbildung 7: Rasterisierung

Schritt 4: Bildberechnung

Im weiteren Verlauf wird für jede der Rasterzellen ein Farbwert berechnet. Die gesamte, aus Rasterzellen bestehende Matrix, wird dabei als Bild oder "Frame" bezeichnet. Statt von Rasterzellen wird in der Praxis häufig von Bildelementen oder Pixeln (engl. "picture elements", kurz pixels) gesprochen.
Der Farb- und Helligkeitswert jedes Pixels hängt von der Farbe des an dieser Position sichtbaren Primitivs ab. Verdeckt liegende Primitive bleiben dabei unberücksichtigt. Zudem bieten die meisten 3D-Rendering-Systeme die Möglichkeit, Lichtquellen in der 3D-Welt zu positionieren, so dass die Primitive in Abhängigkeit von ihrer räumlichen Lage unterschiedlich hell beleuchtet werden. Die verschiedenen Rendering-Verfahren unterscheiden sich unter anderem darin, welche verschiedenen Beleuchtungsmöglichkeiten verfügbar sind.

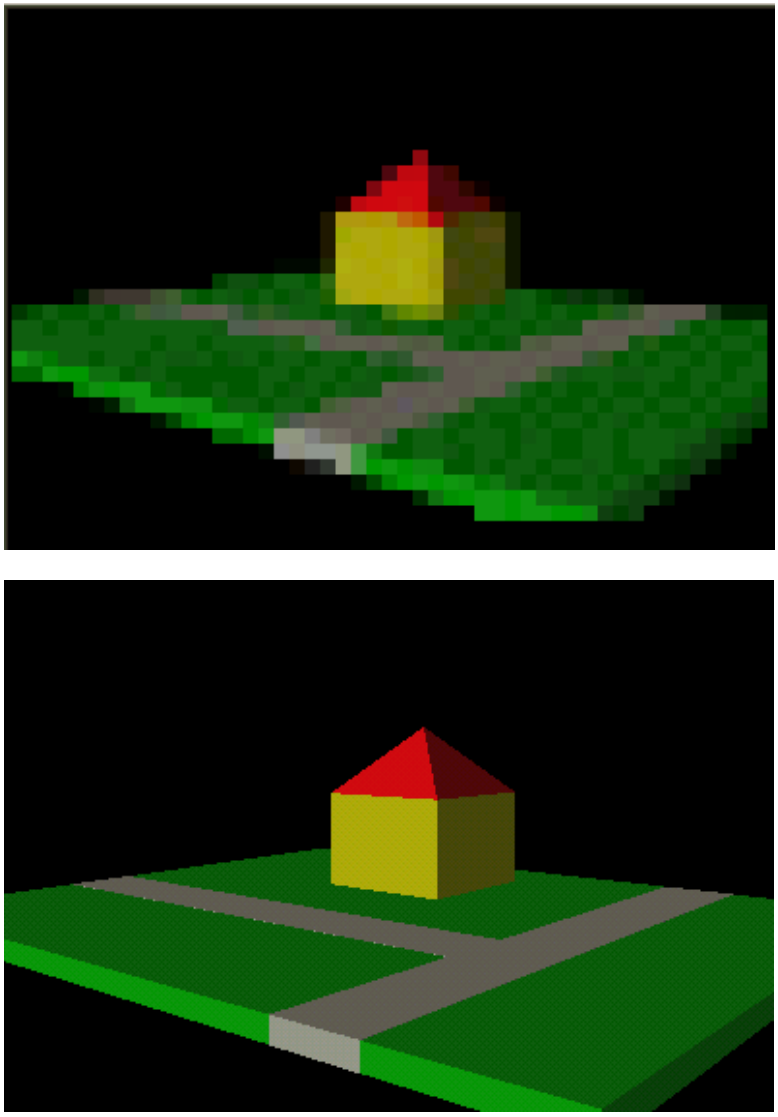

Abbildung 8: Vergleich der Ergebnisse eines groben Rasters (oben) und eines feinen Rasters (unten)

Für das grobe Raster aus Schritt 3 ergibt sich das obere der beiden Bilder. Praktisch wird eine feinere Auflösung gewählt (im Beispiel 398 x 275 Pixel), die das untere Bild zeigt:

Je nachdem, ob die Bildberechung CPU-basiert erfolgt oder ob spezielle 3D-Grafik-Hardware verwendet wird, sind Software- und Hardware-Rendering zu unterscheiden. In letzterem Fall lassen sich 3D-Anwendungen mit der Fähigkeit zu einem **Echtzeit-Rendering** umsetzen, das heißt mit der Fähigkeit, Bilder in Sekunden-Bruchteilen zu berechnen (sogenannte "immediate mode renderer"). Werden ca. 15 Bilder pro Sekunde ("frames per second", kurz fps) dargestellt, entsteht für den Menschen der Eindruck eines "fließenden" Bildes. Diese Eigenschaft ist insbesondere dann bedeutsam, wenn die Anwendung einen hohen Interaktivitätsgrad aufweisen soll (zum Beispiel ruckfreies Fortbewegen durch virtuelle Räume oder Manipulation von Objekten direkt in der Szene). Software-Rendering ist beispielsweise dann von Bedeutung, wenn zur Berechnung fotorealistischer Bilder eine größere algorithmische Freiheit für die getreuere Nachbildung von Beleuchtungssituationen benötigt wird ("high-quality renderer"). Die Bildberechnungszeiten können in diesem Fall sehr lang werden, die Echtzeit-Eigenschaft ist im Regelfall nicht gegeben.

Schattierungsverfahren

Durch 3D-Schattierungsverfahren lassen sich gekrümmte Oberflächen in Abhängigkeit von der Beleuchtungssituation plastisch darstellen. Für die Schattierung werden die Oberflächen in Primitive zerlegt und pixelweise das zugehörige Bild berechnet. Für die folgenden Ausführungen werden als Primitive Dreiecke angenommen. Die verschiedenen Schattierungsverfahren unterscheiden sich darin, wie genau die Farbverläufe innerhalb dieser Primitive dargestellt werden. Die wichtigsten Schattierungsverfahren sind:

- Flat-Shading: Alle innerhalb eines Dreiecks liegenden Pixel werden gleich schattiert.

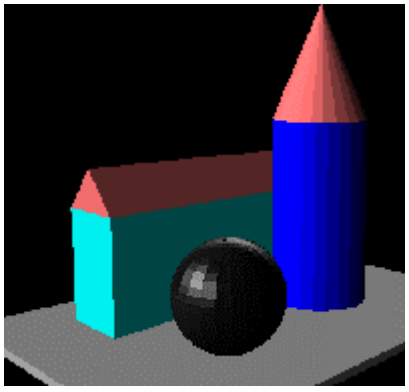

Abbildung 9: Flat-Shading

- Gouraud-Shading: Bei diesem 1971 von Henri Gouraud vorgestellten Verfahren wird die Farbe für jede Ecke des eines Dreiecks berechnet, dazwischen erfolgt für alle Pixel eine lineare Interpolation.

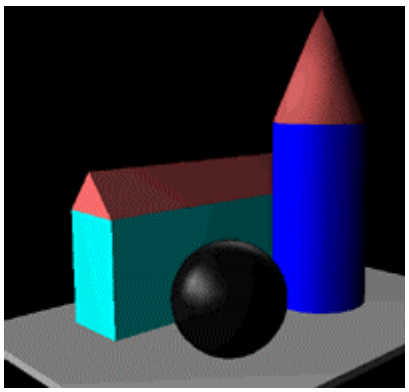

Abbildung 10: Gouraud-Shading

- Phong-Shading: Beim Phong-Shading werden die Normalen-Vektoren (d. h., auf der Oberfläche lotrecht stehende Vektoren, welche die Ausrichtung der Oberfläche im 3D-Raum beschreiben) interpoliert. Dadurch sind neben weichen Farbverläufen auch "Glanzpunkte" möglich, was zu einer realistischeren Darstellung führt.

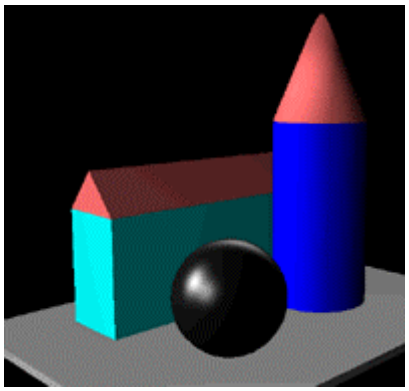

Abbildung 11: Phong-Shading

- Ray-tracing: Beim Ray-tracing werden für die Bild-Berechnung Lichtstrahlen verfolgt. Hierdurch lassen sich Schattenwurf, Lichtbrechungen und -spiegelungen berücksichtigen. Im Vergleich zu den vorgenannten Verfahren ist die Berechnung sehr aufwändig).

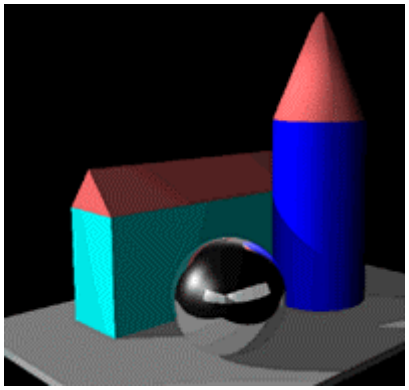

Abbildung 12: Ray-Tracing

- Radiosity: Bei diesem Verfahren werden die physikalischen Verhältnisse der Beleuchtungssituation modelliert, wodurch sich weitergehende Effekte wie diffuse Beleuchtung und Schlagschatten berechnen lassen. Das Verfahren ist sehr Rechenzeit-aufwändig.

In OpenGL-basierten Systemen werden lediglich die beiden erstgenannten Verfahren unterstützt. D. h., eine Schattenwurf-Berechnung ist mittels der OpenGL nicht möglich. Aufwändigere Verfahren wie Ray-Tracing und Radiosity sind im Regelfall heute nicht in Echtzeit durchführbar, d. h. für den Aufbau navigierbarer Umgebungen nur sehr bedingt geeignet.

Von Bedeutung sind sie allerdings für die Erstellung nicht-interaktiver fotorealistischer Darstellungen. Von zentraler praktischer Bedeutung ist die Technik des Texture-mapping, welche die Projektion von Bitmaps auf Oberflächen erlaubt. Auf Grund der gegebenen Hardware-Unterstützung lässt sich diese Technik auch auf PCs für die Generierung plastisch wirkender Darstellungen nutzen. Es handelt sich dabei weniger um ein Schattierungsverfahren, als vielmehr um eine Technik, die sich in Verbindung mit den oben genannten Verfahren einsetzen lässt.

Koordinatensysteme

Zur Positionierung grafischer Objekte werden unterschiedliche Koordinaten-Systeme verwendet, z.B.

- Geo-Koordinaten beschreiben die Objektpositionen bezogen auf die Erdoberfläche; Beispiele: geografische Länge und Breite, Gauß-Krüger-Koordinaten, UTM-Koordinaten in Ergänzung mit einem Höhenwert ü. NN
- lokale Geo-Koordinatensysteme, z. B. unter Verwendung eines gedrehten Koordinatensystems, dessen Nullpunkt im betrachteten Bereich liegt.
- Geräte-Koordinaten beschreiben die Lage der grafischen Objekte auf dem spezifischen Ausgabegerät
  Beispiele: Bildschirm-Koordinaten als Anzahl der Bildpunkte in Zeilen- und Spaltenrichtung; Lage eines vom Laserdrucker gesetzten schwarzen Punktes auf einer DIN A4-Seite.
- Koordinatensystem des 3D-Darstellungsraums: Koordinatensystem der dargestellten 3D-Welt.

Shape-Spezifikationen erfordern Koordinatenangaben. In der Computer-Grafik wird die Horizontale des 2D-Ausgabemediums zumeist als X-Richtung (nach rechts) und die Vertikale als Y-Richtung (nach oben) ausgewiesen. Die positive Z-Richtung zeigt in Richtung des Betrachters (aus dem Bildschirm heraus). Durch diese im Weiteren durch Großbuchstaben kenntlich gemachten Achsen wird ein rechthändiges Koordinatensystem definiert.

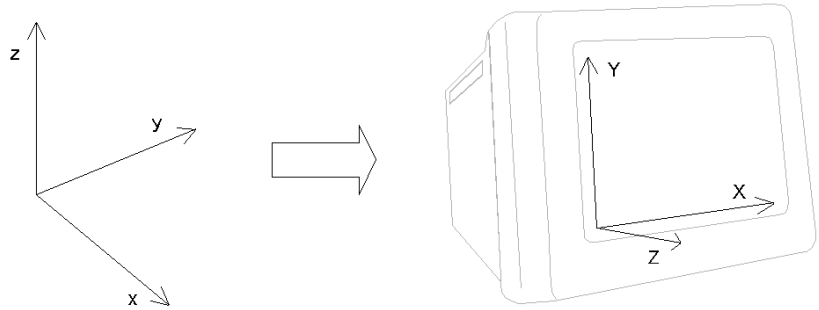

Abbildung 13: Koordinatensystem und 2D-Ausgabemedium

Die Geoobjekt-Koordinaten x und y beziehen sich zumeist auf eine Position auf der Erdoberfläche, z gibt die Höhe über einer Bezugsfläche an. Da im Darstellungsraum die Horizontale als Bezugsebene dient, werden bei der Umrechnung der Geokoordinaten in die Koordinaten des Darstellungsraums die Achsen oft wie folgt einander zugewiesen:

X = x

Y = z

Z = -y

Davon abweichende Zuordnungen sind möglich. In der für den Endanwender konzipierten Software ist diese Unterscheidung nicht zu beachten. Sie betrifft allerdings die Software-Entwickler von 3D-Geovisualisierungssystemen.

Für die Umrechnung zwischen den verschiedenen Bezugssystemen müssen entsprechende Transformationsfunktionen verwendet werden. Zu den weiteren wichtigen Umrechnungen gehören die Verschiebung, Skalierung und Rotation von Objekten. Im 3D-Raum sind darüber hinaus die Projektion von 3D-Koordinaten in 2D-Ebenen sowie die Berechnung perspektivischer 3D-Ansichten von besonderer Bedeutung.

Farbmodelle

Farbe ist heute ein wesentliches Element in der Computer-Grafik. Die Farbtiefe (d. h. die farbliche Differenzierung) kann dabei sehr unterschiedlich sein, z.B.:

- 4-bit-Darstellung in 16 Farben reicht für einfache Grafiken aus
- 8-bit-Darstellung in 256 Farben wird für anspruchsvollere Grafiken und einfache Farbbilder benötigt
- 24-bit-Darstellung in ca. 16,7 Millionen Farben wird für sogenannte True-colour-Bilder benötigt.

Für die Erzeugung unterschiedlicher Farben gibt es verschiedene Farbmodelle. Das gebräuchlichste ist das RGB-Farbmodell, bei dem die Farben aus der additiven Überlagerung der drei Grundfarben Rot, Grün und Blau erzeugt werden können.

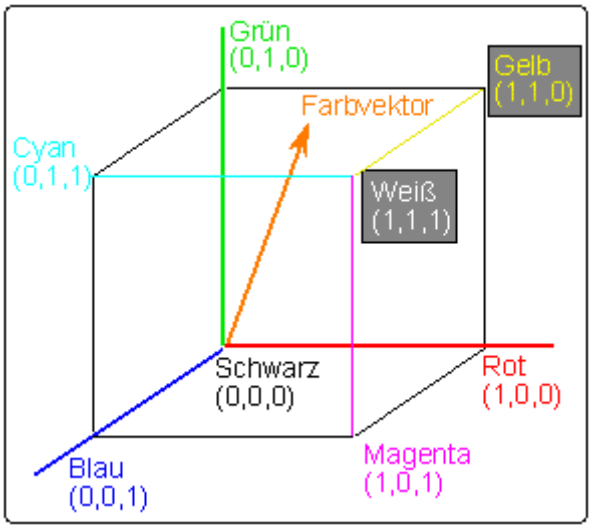

Abbildung 14: RGB-Farbwürfel

Das RGB-Modell lässt sich sehr einfach mit dem eingeblendeten Farbwürfel (Abbildung 14) veranschaulichen. Die drei Primärfarben Rot, Grün und Blau bilden die Hauptachsen (Koordinatenachsen) des Farbwürfels. Der Farbanteil jeder Primärfarbe kann zwischen 0 (nicht vorhanden) bis 1 (volle Farbsättigung) reichen. Innerhalb dieses Intervalls [0, 1] sind theoretisch alle Werte möglich. Jedem Punkt (Farbvektor) im Farbwürfel wird genau eine Farbkombination (r, g, b) zugeordnet .
Praktisch ist die Anzahl der Farben durch die Leistungsfähigkeit der Grafikeinheit des Computers beschränkt; im Falle von True-colour ist das reellzahlige Intervall [0, 1] z. B. auf ganzzahlige Werte [0, 255] pro Grundfarbe abgebildet; RGB(255, 255, 255) entspricht dann der Farbe Weiß.

Das dazu duale (subtraktive) Modell ist das CMY-Farbmodell, das von den Komplementärfarben Cyan, Magenta und Yellow ausgeht.

Alternativ kann man Farbton (Hue), Farbsättigung (Saturation) und einen Helligkeitswert (Lightness) verwenden und erhält dann das HSL-Farbmodell. Diese drei Farbmodelle sind hier vergleichend dargestellt:

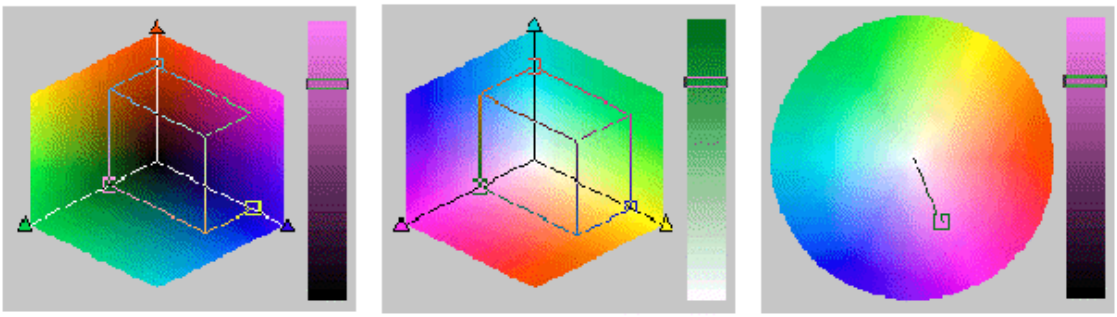

Abbildung 15: Verschiedene Farbmodelle

Neben den eigentlichen Farbwerten lässt sich die Information über den Transparenzgrad mitführen, z. B., indem Farben als (r, g, b, a)-Werte verwaltet werden. Ein Alpha-Wert a von 1 steht hierbei für einen Transparenzgrad von 0 % (gleichbedeutend mit einer Opazität von 100 %), a = 0 für eine Transparenz von 100 % (Opazität = 0 %).

Den Effekt, der mittels des sogenannten Alpha-Blendings erzielbar ist, zeigt Abbildung 16.

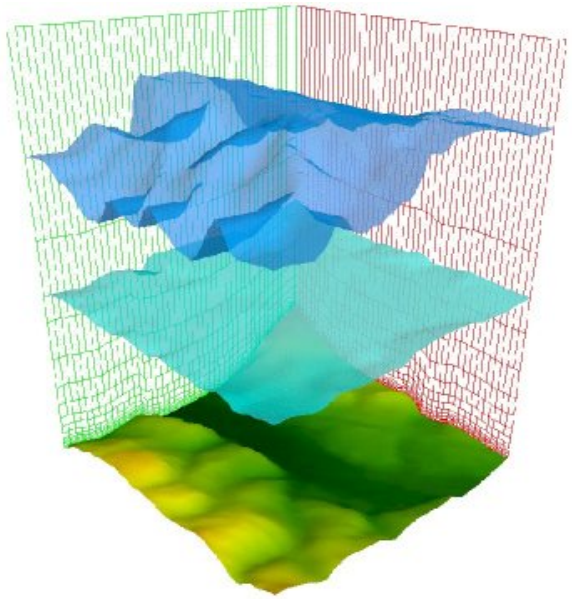

Abbildung 16: Alpha-Blending

Szenengraph-Modellierung

Eine Szene umfasst sämtliche für das Rendering benötigte Information wie z. B. die geometrischen Beschreibungen der Visualisierungsobjekte, visuelle Attribute (z. B. Farben), Beleuchtungssituation und Kameraeinstellungen. Das Szenengraph-Modell ist heute das verbreitetste Paradigma für die Beschreibung dreidimensionaler Szenen. Das Hauptaugenmerk der Szenengraph-Modellierung gilt insbesondere dem Weg zur bildhaften Darstellung.

Neben dem eigentlichen Rendering ermöglicht die Szenengraph-Modell die Definition und Verwaltung komplexer Visualisierungsobjekte sowie des interaktive und – insofern benötigt – zeitlichen Verhalten der Szene.

Bibliotheken, welche die Anordnung von Objekten in Szenengraphen ermöglichen, sind Open Inventor (Silicon Graphics), die im World Wide Web stark verbreitete Virtual Reality Modelling Language VRML oder Java 3D (Sun).

Bei den Szenengraphen handelt es sich um gerichtete, azyklische Graphen ("directed acyclic graphs", kurz DAGs). Häufig wird von Baum-Strukturen statt DAGs gesprochen. DAG und Baum-Struktur unterscheiden sich im Umfeld der Szenengraph-Modellierung dadurch, dass in letzterem Fall kein Kindknoten mehr als einen Elternknoten besitzen darf. Die Knoten des Szenengraphen sind mit Rendering-Objekten wie Shapes(s. Kap. „Geometrische Primitive“) oder geometrischen und grafischen Attributen verknüpft und organisieren diese hierarchisch.

- Grafische Attribute: Über grafische Attribute lässt sich die visuelle Erscheinung ("appearance") der Shapes steuern. Beispielhaft seien Angaben zur Oberflächencharakteristik (Farben, Texturen, Reflektionskoeffizienten, Lichtemissionseigenschaften, etc.) oder Angaben zur Repräsentationsqualität (Tesselationstiefe, Shading-Verfahren, Linienbreiten in Pixeln etc.) genannt.
- Geometrische Attribute: Hierdurch lassen sich Objektgeometrien manipulieren. Von Bedeutung sind insbesondere geometrische Transformationen. Hierunter fallen Translationen, Skalierungen und Rotationen ebenso wie Objekte, welche die Blick- oder Bewegungsrichtung der Kamera liefern.

Durch die Topologie des Graphen lassen sich auf einfache Weise Attribute spezifizieren, die sich auf mehrere Teilgraphen beziehen sollen. Hierunter fallen zum Beispiel die Festlegung eines gemeinsamen Erscheinungsbildes für mehrere Visualisierungsobjekte oder die Spezifikation lokaler Koordinatensysteme. Mit dem Szenengraphen zu verknüpfen sind eine Kamera, welche einen sichtbaren Raumausschnitt liefert, und eine Beleuchtungssituation.

Eine Szene ist nicht als statische Einheit zu betrachten. Beispielsweise können sich Rendering-Objekte in Abhängigkeit von einer durchgeführten Benutzerinteraktion oder dem Ablaufen der Zeit (Animation) ändern. Die heute verfügbaren Szenengraph-Bibliotheken stellen eine Vielzahl verschiedener Verhaltensklassen zur Verfügung. Zu den wichtigsten Klassen zählen:

- Klassen zur Unterstützung der Kamerasteuerung,
- Klassen zur Handhabung Benutzer-generierter Ereignisse (zum Beispiel über Eingabegeräte wie Maus, Tastatur etc.),
- Picking-Klassen, welche die Ermittlung eines im Bild angeklickten Shapes ermöglichen,
- Level-of-Detail-Klassen, die abhängig vom Abstand des Betrachters bzw. der Kamera von einem Objekt die Darstellung in unterschiedlichen Detailliertheitsgraden ermöglichen.
- Klassen zur Umsetzung des sogenannten Billboard-Verhaltens, durch welches sich Shapes in Richtung des Betrachters ausrichten lassen,
- Klassen zur Kollisionserkennung,
- Klassen zur Unterstützung der Spezifikation zeitlicher Abläufe (Interpolatoren, Morphing).

Beispiel zur Szenengraph-Modellierung

Abbildung 17 zeigt eine Möglichkeit der Spezifikation eines einfachen Baumsymbols ("Lollipop"), das sich aus einem braun eingefärbten Zylinder-Shape und einer um die Strecke s in der Vertikalen verschobenen und in der Höhe um den Faktor 1,5 gestreckten Kugel mit vorgeschalteter grüner Farbgebung zusammensetzt:

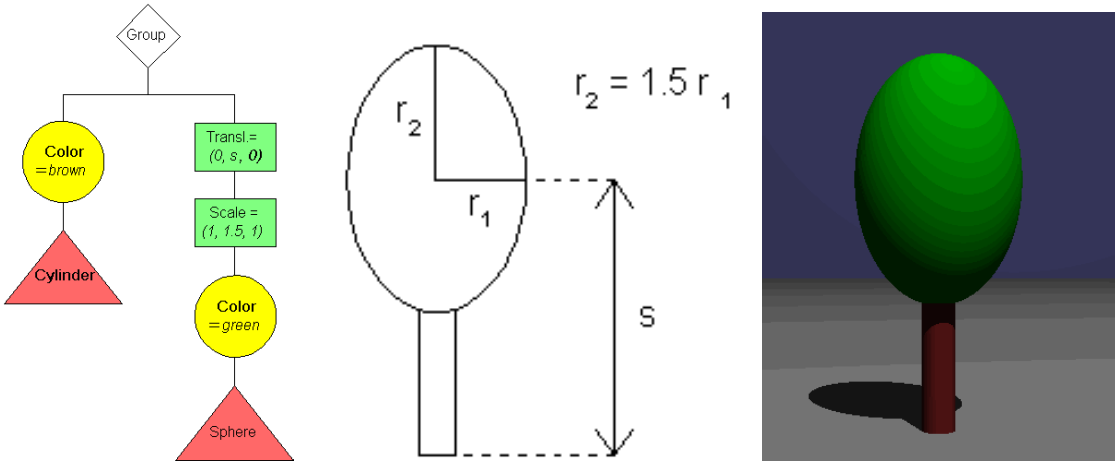

Abbildung 17: Szenengraph, Konstruktionsparameter und gerenderte Darstellung mit zusätzlicher Ebene

Nicht jedes API bietet alle genannten Verhaltensklassen an. Als Vorteile der Verwendung von Szenengraphen sind unter anderem zu nennen:

- die Vereinfachung der Konstruktion komplexer Objekte
- die bessere Speicher-Ausnutzung im Falle mehrfach benutzter Objekte
- die einfachere Bild-Aktualisierung bei Objekt-Änderungen (da nur die Objekte auf den hierarchisch höheren Ebenen aktualisiert werden müssen)

Literatur

(1) Foley, J. D., A. van Dam, S.K. Feiner, J.F. Hughes (1996): Computer Graphics : Principles and Practice. 2nd ed., Reading, MA: Addison-Wesley.

(2) Watt, A. (2000): 3D Computer Graphics. 3rd ed., Harlow, Essex, U.K.: Addison-Wesley.

(3) Bouvier, D. J. (2000): Getting Started with the Java 3D API. Mountain View, CA: Sun Microsystems, http://java.sun.com

(4) Schroeder, W., K. Martin & B. Lorensen (1997): The Visualization Toolkit : An Object-oriented Approach to 3D Graphics. 2nd ed., Upper Saddle River, NJ: Prentice Hall