Computer-Graphik

Helmut Alt, FU Berlin

1 Einleitung

Bekanntlich wurden anfangs Computer dafür entwickelt und eingesetzt, lange und schwierige numerische Rechnungen durchzuführen, danach wurde auch die nichtnumerische elektronische Datenverarbeitung immer wichtiger. Ein- und Ausgabe geschah schriftlich in Form von aus Buchstaben und Zahlen bestehenden Texten, Tabellen usw. In den letzten zwanzig Jahren jedoch hat die Erzeugung von *Bildern* mit Hilfe des Computers, die *Computer-Graphik*, sehr an Bedeutung gewonnen, vor allem deshalb, weil für viele Anwendungen eine bildliche Ausgabe für den Benutzer wesentlich informativer und ausdrucksstärker ist als eine textuelle.
Zu den wichtigsten dieser Anwendungen gehören die unter dem Begriff *CAD* (*"Computer-Aided Design"*) zusammengefaßten. Beim CAD geht es darum, später industriell anzufertigende Objekte, wie z.B. Gebäude, Autokarosserien, Maschinenteile oder elektronische Schaltkreise zunächst mit Hilfe des Computers bildlich zu entwerfen. (Natürlich werden gleichzeitig auch die genauen Maße numerisch abgespeichert.) Dies läßt sich leichter und schneller durchführen als das bisherige manuelle "technische Zeichnen". Vor allem können Änderungen mit Hilfe von Eingabegeräten wie Maus oder Lichtgriffel am Bild direkt innerhalb weniger Minuten vorgenommen werden.
Weitere bedeutende Einsatzgebiete der Computer-Graphik sind die *Flugsimulation*, die die Pilotenausbildung wesentlich vereinfacht hat, die *Computerkunst*, und nicht zuletzt, die wahrscheinlich häufigste Anwendung, die *Videospiele*. Fast alle diese Anwendungen sind *interaktiv*, das heißt beruhen auf einem ständigen Austausch von Signalen zwischen Computer und Benutzer, einige davon erfordern auch die Erzeugung bewegter Bilder (*Computer-Animation*).
Fertige Graphik- und CAD-Softwarepakete sind im Handel erhältlich. In diesem Artikel sollen nicht die Einsatzmöglichkeiten und die Eigenschaften solcher Systeme erläutert werden, sondern vielmehr die theoretischen Grundlagen und wichtigsten Algorithmen und Datenstrukturen zur Herstellung graphischer Software. Neben physikalischen Grundlagen aus der geometrischen Optik werden dabei auch mathematische Methoden und Erkenntnisse, insbesondere aus der darstellenden und der analytischen Geometrie benutzt. Geeignete Algorithmen und Datenstrukturen werden in der *algorithmischen Geometrie* entwickelt, einem Teilgebiet der theoretischen Informatik, das sich in den letzten Jahren, vor allem inspiriert durch Computer-Graphik-Anwendungen, entwickelt hat.
Abschnitt 2 dieses Artikels wird Methoden angeben, wie man Objekte und Szenen des dreidimensionalen Raums formal beschreibt und im Computer abspeichert. Abschnitt 3 behandelt die Sichtbarmachung solcherart gespeicherter Szenen auf

einem zweidimensionalen Ausgabemedium, wie etwa einem Bildschirm. Abschnitt 4 behandelt das Problem von Ausleuchtung und Schattierung, insbesondere auch bei durchsichtigen und reflektierenden Objekten, wenn diese Effekte möglichst realistisch ("photorealistisch") erzielt werden sollen. Tafel 1 zeigt ein mit Mitteln der Computer-Graphik erzeugtes photorealistisches Bild[1].

2 Darstellung dreidimensionaler Objekte

Wir wenden uns zunächst der Frage zu, wie man eine dreidimensionale Szene, wie z.B. die in Tafel 1 dargestellte[2], im Rechner abspeichern kann.
Eine Möglichkeit ist, die auftretenden Objekte aus einer kleinen Menge leicht zu beschreibender Objekte ("Primitive"), wie Quader, Zylinder, Pyramiden, Kegel, Kugeln "zusammenzubauen". Als mengentheoretische Operationen nimmt man dabei etwa "Vereinigung", "Durchschnitt" und "Mengendifferenz". Dieser in der englischsprachigen Literatur "constructive solid geometry" (CSG) genannte Ansatz ist bei technischen Anwendungen, etwa beim CAD, sehr sinnvoll, zumal er auch Hinweise für die Konstruktion der Objekte gibt. Jedoch bei der Darstellung natürlicher Objekte, z.B. Pflanzen oder Gesichtern, oder bei künstlerischen Anwendungen der Computer-Graphik reichen diese einfachen Bausteine nicht aus. Die Form der auftretenden Oberflächen ist wesentlich komplizierter und kann beliebig unregelmäßig werden. Solche "Freiformflächen" können z.B. durch viele kleine regelmäßige Flächenstücke approximiert werden.
Beide Ansätze sollen in diesem Kapitel vorgestellt werden.

2.1 Geometrisches Modellieren

Wie schon erwähnt, ist die Grundidee der CSG, komplizierte Objekte aus einfachen Bausteinen mit Hilfe der Mengenoperationen [3]
$\cup$, $\cap$, $\setminus$ aufzubauen. Wir wollen uns dies an einem Beispiel veranschaulichen und zwar an dem in Abb. 1 dargestellem Objekt.
Die folgenden Bausteine sollen verwendet werden können:
a) $Q\,(\boldsymbol{u},\, \ell_1,\, \ell_2,\, \ell_3) \qquad \boldsymbol{u} \in I\!R^3;\; \ell_1, \ell_2, \ell_3 \in I\!R$
sei der Quader mit $\boldsymbol{u}$ als linkem unteren, vorderen Eckpunkt und Kanten parallel zur x-, y- und z-Achse mit den Seitenlängen ℓ_1, ℓ_2, ℓ_3.
b) $Z(\boldsymbol{u},\, \boldsymbol{v},\, r) \qquad \boldsymbol{u},\, \boldsymbol{v} \in I\!R^3;\; r \in I\!R$
sei der Zylinder vom Radius r, dessen Grundflächen die Mittelpunkte $\boldsymbol{u}, \boldsymbol{v}$ haben.

[1] Tafel 1 wurde nicht in die vorliegende Datei mit aufgenommen sondern befindet sich separat auf den Webseiten der Electronic Library, siehe http://www.emis.de/monographs/schulz/alt.html.

[2] siehe die vorhergehende Fußnote.

[3] Für Mengen A, B ist definiert:
$A \cup B := \{x | x \in A \text{ oder } x \in B\}$, $A \cap B := \{y | y \in A \text{ und } y \in B\}$ und $A \backslash B := \{z \in A | z \notin B\}$.

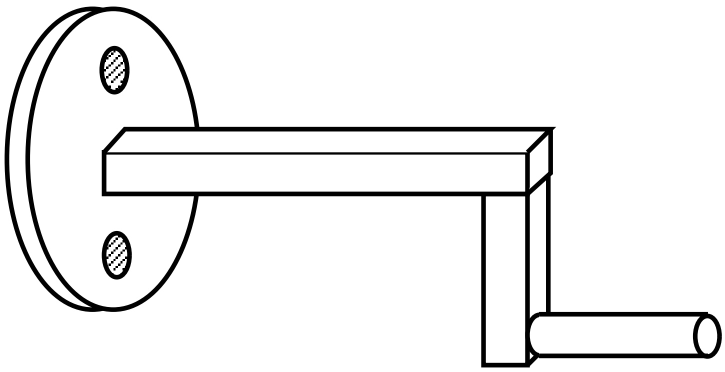

Abbildung 1: Kurbel

Dann läßt sich die Kurbel K aus Abb. 1 z.B. wie folgt als Objekt in $I\!R^3$ beschreiben:

$$\begin{array}{lcll}
S0 & := & Z\left((0,0,0),(1,0,0),7\right) & \text{Scheibe} \\
B1 & := & Z\left((0,0,4),(1,0,4),1\right) & \left.\vphantom{\begin{array}{l}a\\a\end{array}}\right\} \text{Bohrungen} \\
B2 & := & Z\left((0,0,-4),(1,0,-4),1\right) & \\
A & := & Q\left((0,-1,-1),20,2,2\right) & \text{Achse} \\
T & := & Q\left((18,-1,-9),2,2,8\right) & \text{Querstange} \\
H & := & Z\left((20,0,-8),(28,0,-8),1\right) & \text{Handgriff}
\end{array}$$

Damit sind die Einzelteile definiert, das Gesamtobjekt läßt sich aus ihnen mit mengentheoretischen Operationen aufbauen:

$$K := A \cup T \cup H \cup ((S0 \setminus B1) \setminus B2)\,.$$

Oft wird dies auch mit Hilfe eines “Konstruktionsbaumes” beschrieben, der für unser Beispiel in Abb. 2 angegeben ist.
Für einige spezielle dreidimensionale Objekte, die jedoch in der Praxis häufig auftreten, gibt es noch einfachere Methoden des “solid modeling”. So ist z.B. ein *Translationskörper* K die Teilmenge des dreidimensionalen Raumes, die überstrichen wird, wenn man eine ebene Figur F um einen Vektor $\boldsymbol{v}$ verschiebt (siehe Abb. 3). K läßt sich also durch Angabe von $\boldsymbol{v}$ und F darstellen. Dies kann man dadurch verallgemeinern, daß statt einer geradlinigen Bewegung entlang $\boldsymbol{v}$ eine Bewegung entlang einer beliebigen Kurve C möglich ist. Falls C ein Kreis ist, so spricht man von einem *Rotationskörper*, in diesem Fall ist es aber sinnvoller, neben dem zweidimensionalen Objekt F die Rotationsachse a direkt anzugeben (siehe Abb. 4).
Eine allgemeine Methode, bei der dreidimensionale Objekte jedoch nicht exakt dargestellt, sondern nur approximiert werden, ist die Verwendung von sogenannten *Octrees*, d.h. Bäumen, deren innere Knoten acht Nachfolger haben. Die zugrunde liegende Idee ist folgende:

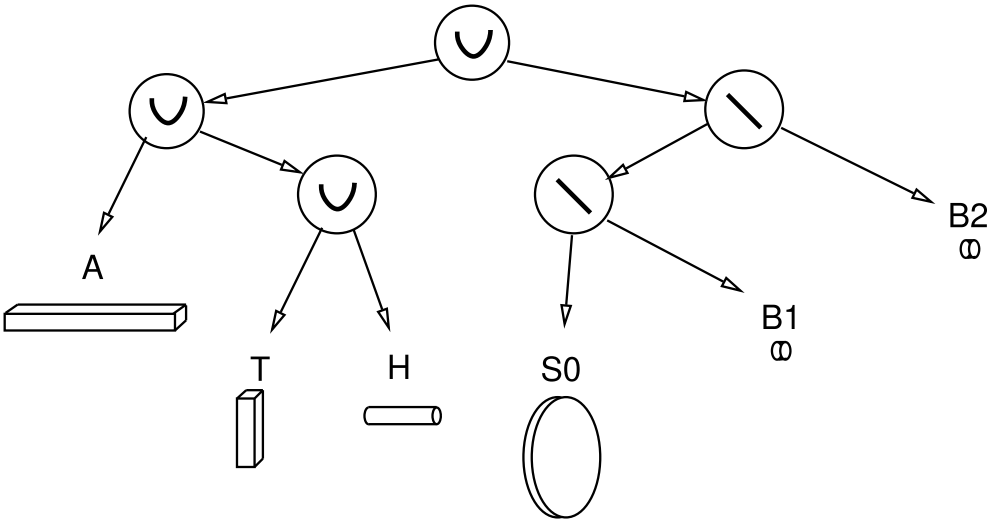

Abbildung 2: CSG-Konstruktionsbaum

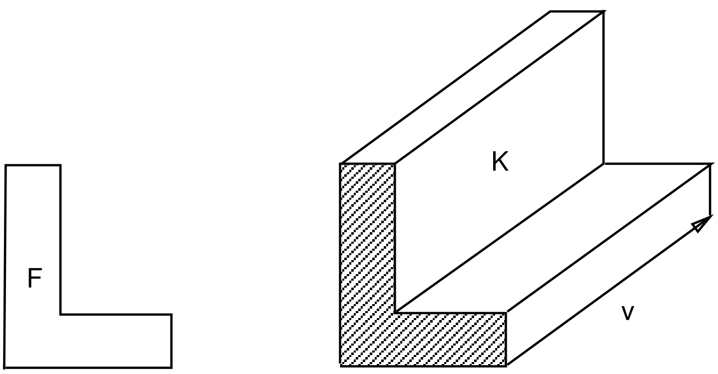

Abbildung 3: Translationskörper

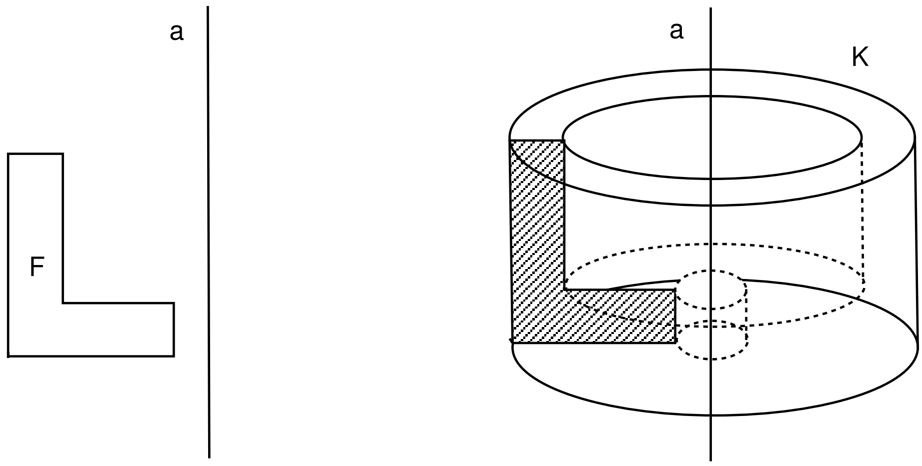

Abbildung 4: Rotationskörper

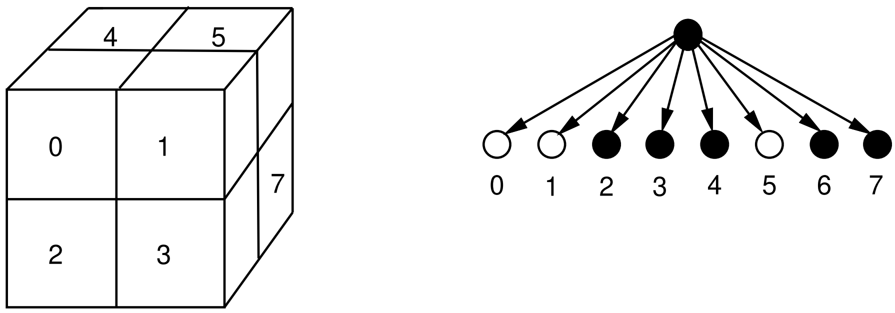

Abbildung 5: Oktanden eines Würfels, zugehöriger Knoten eines Octrees.

Angenommen, das darzustellende Objekt K befinde sich innerhalb eines Würfels W. W gilt als erste Approximation für K. Will man K genauer darstellen, so teilt man W in acht Würfel der halben Seitenlänge, ("Oktanden"), die wie in Abb. 5 durchnumeriert sind.

Diese Aufteilung kann man durch einen Baum mit einer Wurzel und acht Nachfolgern darstellen, so daß die Knoten im Baum verschiedenen Teilwürfeln entsprechen. Für diejenigen, die einen nichtleeren Schnitt mit K haben, sind die entsprechenden Knoten schwarz gefärbt. Die zweite Approximation für K wäre also die Vereinigung der Teilwürfel $2, 3, 4, 6$ und 7. Nun kann man durch wiederholtes Anwenden dieser Methode das dreidimensionale Raster zur Darstellung von K beliebig verfeinern, wodurch der Octree immer höher wird (siehe Abb. 6). Effizient wird die Methode dadurch, daß Teilwürfel die ganz im Innern oder ganz außerhalb von K liegen nicht mehr weiter aufgespaltet werden müssen.

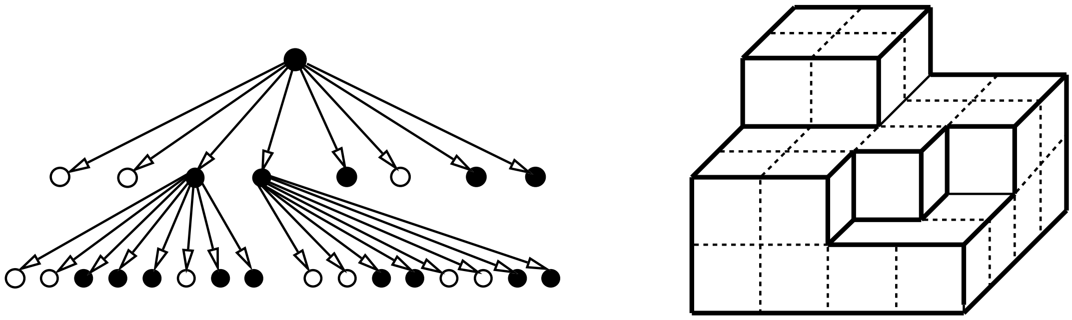

Abbildung 6: Feinere Aufteilung und zugehöriger Octree.

2.2 Freiformflächen

Viele in der Computer-Graphik auftretende Objekte sind nicht regelmäßig genug, um mit den im vorherigen Abschnitt angegebenen Methoden effizient dargestellt werden zu können. Dies gilt insbesondere bei der Modellierung von *natürlichen* Objekten wie z.B. Gesichtern, Pflanzen, Bergen, Wolken, oder wenn eine gewisse künstlerische Freiheit gewünscht ist wie z.B. beim Design von Kleidern, Autokarosserien usw. Da in der Computer-Graphik meist nur die *Oberfläche* dieser Objekte dargestellt werden soll, geht es also um effiziente Modellierung von Flächen im dreidimensionalen Raum.

2.2.1 Polyederflächen

Eine naheliegende und zugleich allgemein verwendbare Art, Oberflächen darzustellen besteht darin, die Fläche mit hinreichend vielen Rasterpunkten zu überziehen und benachbarte Punkte durch gerade Linien (*Kanten*) zu verbinden (siehe Abb. 7 und das Waschbecken[4] in Tafel 1)

Ist das Raster dicht genug, so ergibt diese Darstellung eine realistische Approximation der Oberfläche. Sogenannte *Polyeder*, das heißt dreidimensionale Gebilde, deren Oberfläche wirklich durch ebene Flächen begrenzt ist (z.B. Würfel, Tetraeder), lassen sich natürlich auf diese Weise exakt darstellen. Will man sicherstellen, daß ein von Kanten umrahmtes Flächenstück (*Facette*) in dieser Darstellung *eben* ist, so wählt man die Kanten sinnvollerweise so, daß man ein *dreieckiges* Netz erhält.

Üblicherweise wird eine solche polyedrische Darstellung einer Fläche im Computer so abgespeichert, daß man eine Liste aller Punkte, eine Liste aller Kanten und eine Liste aller Facetten anlegt. Zu jedem Punkt speichert man seine Koordinaten ab. Außerdem legt man Zeiger zwischen den drei Listen an, etwa von jeder

[4] Tafel 1 wurde nicht in die vorliegende Datei mit aufgenommen sondern befindet sich separat auf den Webseiten der Electronic Library, siehe http://www.emis.de/monographs/schulz/alt.html.

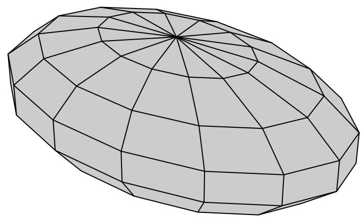

Abbildung 7: Polyederapproximation eines Ellipsoids

Kante zu ihren beiden Endpunkten und zu den beiden anliegenden Facetten und von jeder Facette zu den drei sie begrenzenden Kanten. Dies ermöglicht es zum Beispiel, zu einer gegebenen Facette schnell ihre Nachbarfacetten zu finden und die ganze Oberfläche systematisch Facette für Facette zu durchlaufen, was für viele Algorithmen der Computer-Graphik eine wichtige Teilprozedur ist.
Ein Nachteil der Polyederdarstellung von Flächen ist, daß bei manchen Anwendungen in der Computer-Graphik insbesondere, wenn die Reflexion von darauffallendem Licht simuliert werden soll, sich die Facettenstruktur im Bild bemerkbar macht. Im folgenden sollen deswegen Flächendarstellungen vorgestellt werden, mit denen man auch "glatte" Flächen beschreiben kann.

2.2.2 Parametrisierte Darstellung von Kurven und Flächen

Die in der Mathematik übliche Darstellung von *Flächen* im $\mathbb{R}^3$ und Kurven im $\mathbb{R}^2$ oder $\mathbb{R}^3$ ist die sogenannte *Parameterdarstellung*. Für Kurven ist dies eine stetige Abbildung $\boldsymbol{c} : [0,1] \to \mathbb{R}^3$ (oder $\mathbb{R}^2$) [5], also jede reelle Zahl t aus dem Einheitsintervall $[0,1]$ wird auf einen Punkt $\boldsymbol{c}(t) = (x(t), y(t), z(t))$ (oder $(x(t), y(t))$) abgebildet. Zum Beispiel liefert die Parametrisierung $x(t) = a\cos(2\pi t), y(t) = b\sin(2\pi t)$, für $t \in [0,1]$ in $\mathbb{R}^2$ die Ellipse mit dem Nullpunkt als Mittelpunkt und den Halbachsen a, b (siehe Abb. 8a). Die Parametrisierung $x(t) = \cos(6\pi t), y(t) = \sin(6\pi t), z(t) = t$ liefert die in Abb. 8b dargestellte Schraubenlinie im $\mathbb{R}^3$.
Parameterdarstellungen von *Flächen* sind definiert als stetige Abbildungen der Form $\boldsymbol{c} : [0,1] \times [0,1] \to \mathbb{R}^3$, d.h. jedes Paar (u, v) aus dem "Einheitsquadrat" wird abgebildet in einen Punkt $\boldsymbol{c}(u, v) = (x(u, v), y(u, v), z(u, v))$. Zum Beispiel liefert die Parameterdarstellung

$$x(u, v) \;=\; u\sin(2\pi v)$$

[5] Dabei bezeichnet $f : A \to B$ eine Abbbildung von A in B und $[0,1] := \{x \in \mathbb{R} | 0 \leq x \leq 1\}$

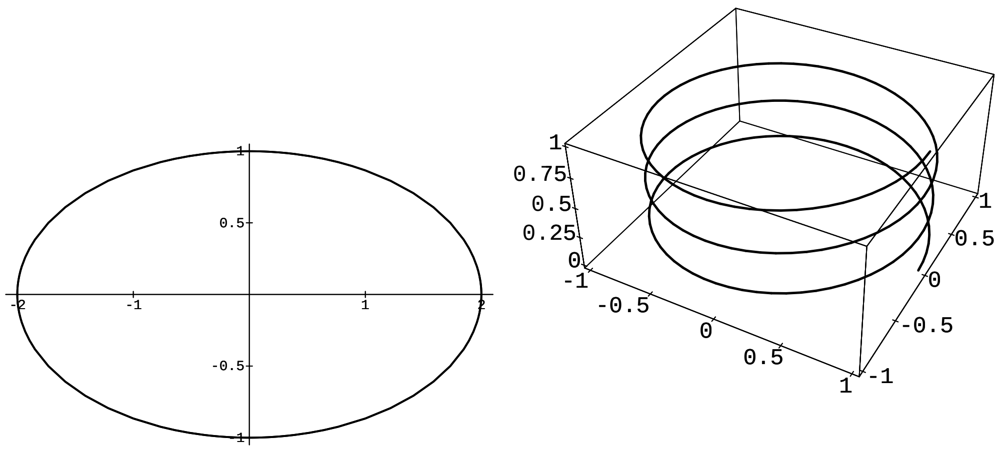

Abbildung 8: a) Ellipse b) Schraubenlinie

$$\begin{aligned} y(u,v) &= u\cos(2\pi v) \\ z(u,v) &= 1/2\sin(2\pi u) \end{aligned}$$

die in Abb. 9 dargestellte Fläche.

2.2.3 B-Splines

Bei Anwendungen in der Computer-Graphik werden parametrisierte Kurven und Flächen oft als Linearkombination einer festen Menge von sogenannten *Basisfunktionen* $B_1, \ldots, B_n$ von $[0,1]$ nach $[0,1]$ dargestellt. Eine Kurve dieser Art in $\mathbb{R}^2$ hat also die Form

$$\boldsymbol{c}(t) = \sum_{i=1}^{n} \boldsymbol{p}_i B_i(t) \tag{2.2.1}$$

d.h. die Kurve ist durch Angabe der Koeffizienten $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n \in \mathbb{R}^2$ eindeutig spezifiziert. Welche Eigenschaften die so darstellbaren Kurven haben, hängt natürlich von der Wahl der Basisfunktionen ab. Eine spezielle Menge von Basisfunktionen, die rekursiv definiert werden, liefert als mögliche Kurven die sogenannten *B-Splines*. Sie sind von großem Interesse im CAD und sollen deswegen im folgenden kurz beschrieben werden:
Angenommen, die Anwendung erfordert es, daß die Kurven der Form (2.2.1)

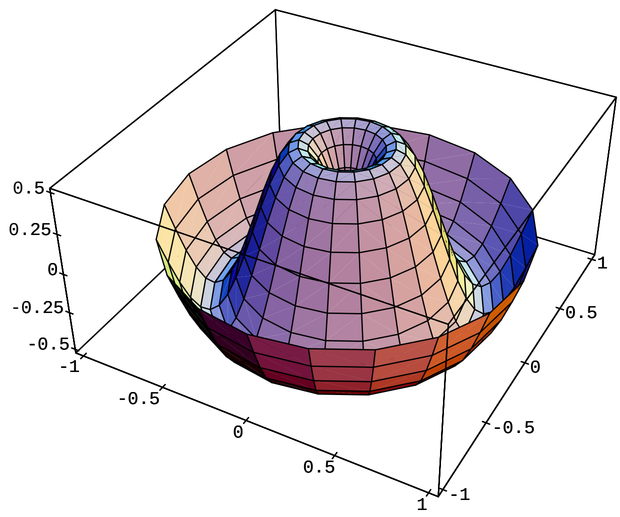

Abbildung 9: Parametrische Fläche

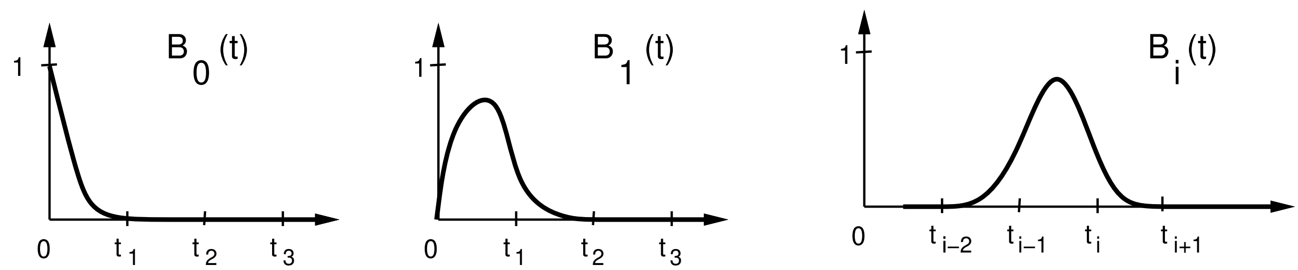

Abbildung 10: Basisfunktionen$B_0(t)$,$B_1(t)$ und $B_i(t)$, $2 \leq i \leq n-2$ für B-Splines 2. Ordnung. B_{n-1}, B_n sind symmetrisch zu B_1, B_0.

"hinreichend glatt" sind, also zumindest stetig differenzierbar.[6] Dies wird dadurch erreicht, daß man eine Basis $B_0, \ldots, B_n$ auswählt, die bereits diese Eigenschaft hat. Nehmen wir dazu an, daß das Einheitsintervall $[0, 1]$ in n gleiche Teile $[t_0, t_1], \ldots, [t_{n-1}, t_n]$ mit $t_i = i/n$ für $i = 0, 1, \ldots, n$, aufgeteilt ist. Die Basisfunktionen B_i haben nun die in Abb. 10 angedeutete Form. B_i $(2 \leq i \leq n-2)$ ist nur auf dem Intervall $[t_{i-2}, t_{i+1}]$ von Null verschieden und hat ihr Maximum an der Stelle $(t_{i-1} + t_i)/2$. Dort ist auch der "Einfluß" der anderen Basisfunktionen relativ gering, ohnehin haben nur B_{i-1} und B_{i+1} dort einen von Null verschiedenen Wert. Dies bedeutet, daß eine Kurve der Form (2.2.1) sich für diesen Parameter dem durch den Koeffizienten $\boldsymbol{p}_i$ gegebenen Punkt im $\mathbb{R}^2$ nähert. Aus diesem

[6]Eine Funktion f heißt stetig differenzierbar, wenn die Ableitung f' existiert und stetig ist.

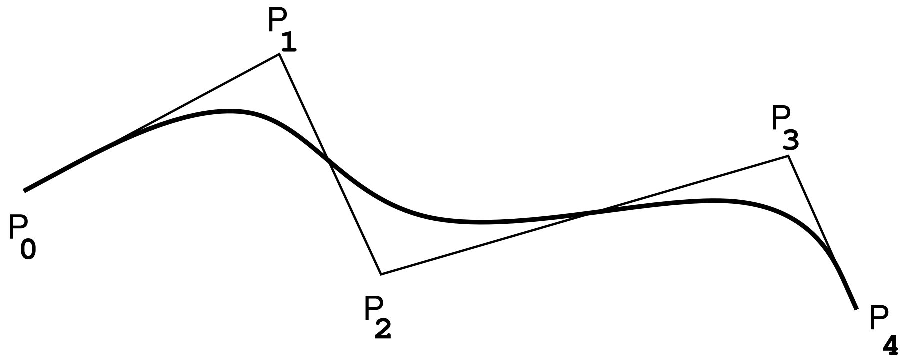

Abbildung 11: B-Spline mit Stützpunkten

Grund bezeichnet man die Punkte $\boldsymbol{p}_0, \ldots, \boldsymbol{p}_n$ auch als *Stützpunkte* des B-Splines $\boldsymbol{c}$. Abb. 11 illustriert an einem Beispiel den Einfluß der Stützpunkte auf den Verlauf der Kurve.

Verändert man den Stützpunkt $\boldsymbol{p}_i$, so ändert die Kurve *lokal* ihren Verlauf im Bereich zwischen $\boldsymbol{p}_{i-2}$ und $\boldsymbol{p}_{i+1}$. Beim Einsatz von B-Splines etwa im CAD ermöglicht dies dem Designer, nach Belieben die Kurve lokal zu verändern bis ihr Verlauf seinen Vorstellungen entspricht. Entscheidend für Anwendungen ist, daß die Kurve auch unter diesen Manipulationen immer "glatt" bleibt; statt "stetig differenzierbar" läßt sich auch "k-mal stetig differenzierbar" für beliebiges $k \in \mathbb{N}$ erreichen.Auch kann man die Kurve "zwingen", durch einen bestimmten Punkt $\boldsymbol{p} \in \mathbb{R}^2$ hindurchzugehen, indem man für das gewünschte i die Stützpunkte $\boldsymbol{p}_i = \boldsymbol{p}_{i+1} = \boldsymbol{p}_{i+2} = \boldsymbol{p}$ setzt. Die Glattheit der Kurve geht dabei allerdings an dieser Stelle verloren. Die Basisfunktionen für solche B-Splines k-ter Ordnung haben eine ähnliche Form wie die in Abb. 10, sind aber auf $k+1$ Teilintervallen von Null verschieden.

Die Idee der B-Splines läßt sich leicht von Kurven in $\mathbb{R}^2$ auf Flächen im $\mathbb{R}^3$ verallgemeinern. Man braucht noch nicht einmal neue Basisfunktionen zu definieren, sondern kommt mit denjenigen für Kurven aus. Eine Fläche $\boldsymbol{c} : [0,1]\times[0,1] \to \mathbb{R}^3$ wird dargestellt in der Form

$$\text{(2.2.2)} \qquad \boldsymbol{c}(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{n} \boldsymbol{p}_{ij} B_i(u) B_j(v)$$

Die Fläche wird bestimmt durch die Koeffizienten $\boldsymbol{p}_{ij} \in \mathbb{R}^3$, $1 \le i,j \le n$, diese bilden ein $m \times n$-"Netz" von Stützpunkten in $\mathbb{R}^3$. Abbildung 12 zeigt ein Beispiel eines solchen Netzes mit der dazugehörigen Fläche.

B-Spline-Flächen können auf ähnliche Art manipuliert werden wie B-Spline-Kurven, und sie sind daher von großer Bedeutung im CAD. Durch einen ähnlichen Ansatz werden die sogenannten *Bézier-Kurven* und *-Flächen* definiert, die

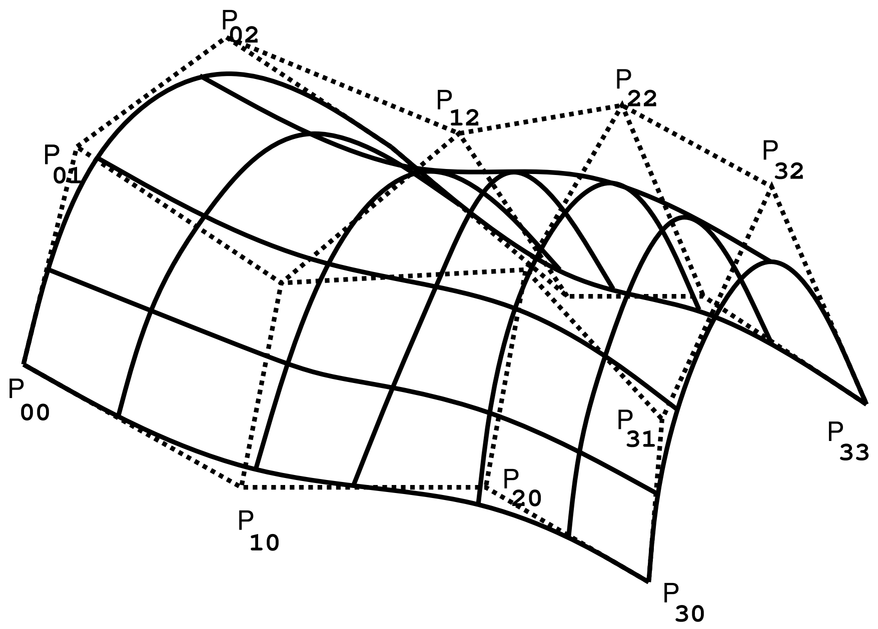

Abbildung 12: B-Spline-Fläche mit Stützpunkten

von Pierre Bézier beim Autohersteller Renault zum Design von Autokarosserien entwickelt wurden. Auch das Waschbecken in Tafel 1 wurde mit B-Spline-Flächen modelliert[7].

3 Wiedergabe dreidimensionaler Szenen

Hat man eine dreidimensionale Szene auf eine der im vorigen Abschnitt beschriebenen Arten im Computer abgespeichert, so ist das nächste wichtige Problem, wie man sie dem Benutzer sichtbar macht. Das geschieht meist über einen Bildschirm, möglicherweise auch über einen Plotter. Jedenfalls muß ein zweidimensionales Bild der dreidimensionalen Szene erzeugt werden.
Natürlich ist diese Problematik, da sie Gegenstand von Künsten und Wissenschaften wie der Malerei, des technischen Zeichnens oder der Kartographie ist, schon seit langer Zeit ausgiebig theoretisch untersucht worden. Die entsprechende mathematische Disziplin ist die darstellende Geometrie. Algorithmisch weitaus schwieriger als die reine Projektion einer dreidimensionalen Szene in die Bildebene ist das Auffinden und Entfernen von Kanten und Flächen, die von anderen Objekten der Szene verdeckt werden und deswegen bei der zweidimensionalen

[7]Tafel 1 wurde nicht in die vorliegende Datei mit aufgenommen sondern befindet sich separat auf den Webseiten der Electronic Library, siehe http://www.emis.de/monographs/schulz/alt.html.

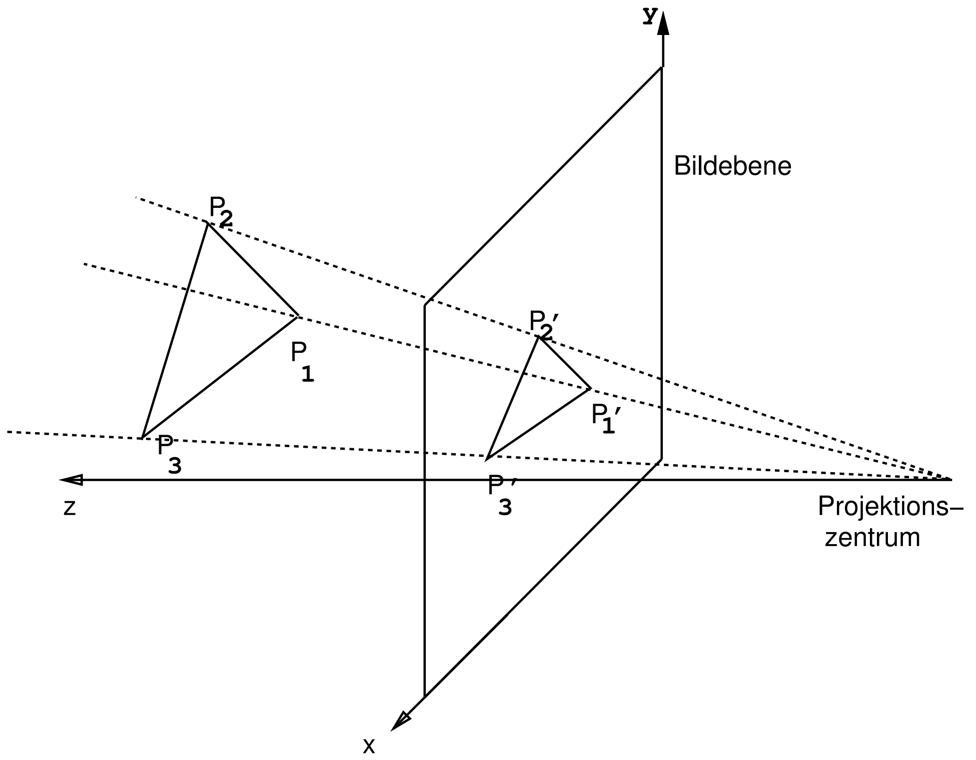

Abbildung 13: Zentralprojektion des Dreiecks $P_1P_2P_3$

Wiedergabe nicht erscheinen sollen.

3.1 Projektionen in die Bildebene

Eine sehr gebräuchliche Abbildung des dreidimensionalen Raumes in die Bildebene ist die *Zentralprojektion*, weil ihre Abbildungseigenschaften denen des menschlichen Auges oder einer Kamera ähnlich sind. Sie hat die Eigenschaft, daß näherliegende Gegenstände größer, weiter entfernt liegende kleiner abgebildet werden. Dabei werden ein Punkt im Raum, das *Projektionszentrum* (*Augpunkt*) und eine Ebene, die *Bildebene* festgelegt. Ein beliebiger Punkt P, der vom Projektionszentrum aus gesehen im Halbraum jenseits der Bildebene liegt, wird nun auf diese Weise abgebildet, indem man ihn durch einen Strahl (*Projektionsstrahl* oder *Sehstrahl*) mit dem Projektionspunkt verbindet. Der Schnittpunkt P' des Projektionsstrahls mit der Bildebene ist der zu P gehörige Bildpunkt (siehe Abb. 13).
Einfachheitshalber wählt man das Koordinatensystem so (wie in Abb. 13 angedeutet), daß die Bildebene die x-y-Ebene ist und das Projektionszentrum im Punkt $(0, 0, -a)$ der negativen z-Achse liegt. Nun kann man mit Hilfe des

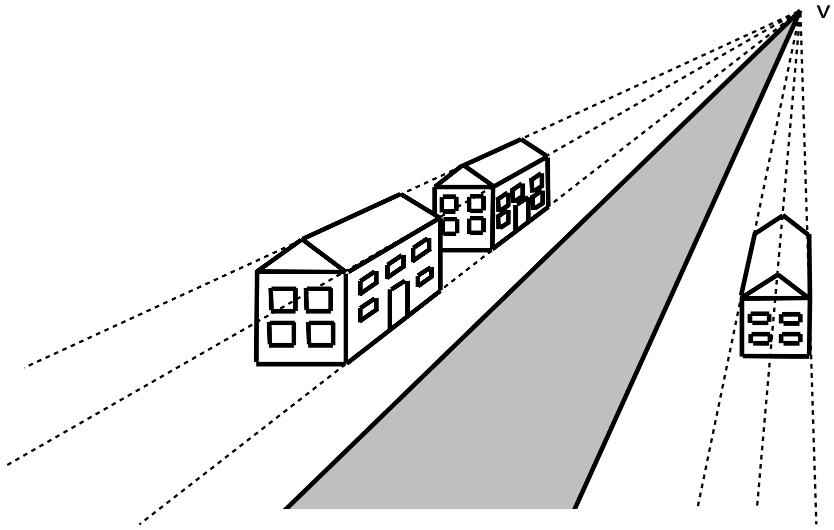

Abbildung 14: Zentralprojektion, v Verschwindungspunkt

Strahlensatzes leicht ausrechnen, daß ein Punkt $P \in I\!R^3$ mit den Koordinaten (P_x, P_y, P_z), $P_z \geq 0$ auf den Punkt P' mit Koordinaten

(3.1.3)
$$\begin{aligned} P'_x &= \frac{a}{P_z + a} \cdot P_x \\ P'_y &= \frac{a}{P_z + a} \cdot P_y \end{aligned}$$

abgebildet wird. Die Tatsache, daß P_z im Nenner steht, bewirkt auch, daß Objekte, die weit von der Bildebene entfernt sind, also große z-Koordinaten haben, entsprechend kleiner abgebildet werden. Auch parallele Geraden g_1 und g_2, sofern sie nicht parallel zur x- y-Ebene sind, werden nicht wieder auf parallele Geraden abgebildet sondern auf Geraden g'_1, g'_2, die sich immer weiter nähern je größer die z-Koordinate auf g_1, g_2 wird. Sie berühren sich schließlich in einem Punkt, dem sogenannten *Verschwindungspunkt*. Abb. 14 zeigt die Zentralprojektion einer dreidimensionalen Szene.
Läßt man das Projektionszentrum auf der negativen z-Achse gegen unendlich gehen, so werden alle Sehstrahlen parallel und senkrecht zur Bildebene. Diese Projektion heißt *Parallelprojektion*. Für $a \to \infty$ entsteht aus (3.1.3) einfach

(3.1.4)
$$\begin{aligned} P'_x &= P_x \\ P'_y &= P_y \end{aligned}$$

das heißt, diese Parallelprojektion ist einfach die Projektion auf die x-y-Ebene. Sind die Kanten der Objekte der Szene parallel zu den Koordinatenachsen, was

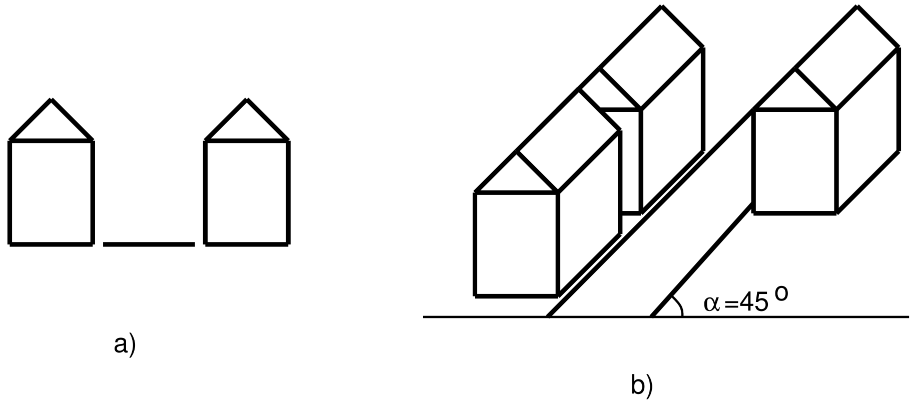

Abbildung 15: Parallelprojektionen; a) orthogonale b) Kavaliersperspektive

bei rechtwinkligen Objekten oft einfachheitshalber so gewählt wird, so hat die Projektion (3.1.4) den Nachteil, daß nur eine Seite der Objekte im Bild sichtbar wird (siehe Abb. 15a). Um einen besseren visuellen Eindruck zu erzielen wählt man daher oft eine *schiefe Parallelprojektion* bei der die Sehstrahlen nicht mehr senkrecht auf der Bildebene stehen. Dadurch können Kanten in allen drei Koordinatenrichtungen sichtbar werden. Die allgemeine Form dieser Projektion ist

$$\begin{aligned} P'_x &= P_x + P_z \cdot d \cdot \cos\alpha \\ P'_y &= P_y + P_z \cdot d \cdot \sin\alpha \end{aligned} \tag{3.1.5}$$

wobei α der Winkel zwischen dem Bild der z-Richtung und der x-Richtung und d der Faktor ist, um die Strecken in z-Richtung verkürzt bzw. verlängert werden (Strecken in x- oder y-Richtung bleiben gleich lang.) Häufig benutzt wird $\alpha = 45°, d = 1/2$, die sogenannte *Kavalierperspektive* (siehe Abb. 15b)
Alle Parallelprojektionen erhalten Parallelität von Geraden, weil sie, im Gegensatz zur Zentralprojektion, lineare Abbildungen sind.
Um also eine im Computer abgespeicherte Szene z.B. auf dem Bildschirm gemäß einer Projektion p der Form (3.1.3) - (3.1.5) wiederzugeben, genügt es, alle Teile (Punkte, Kanten, Oberflächen) mit Hilfe von p in das Bildschirmkoordinatensystem abzubilden. Bei der Darstellung durch CSG genügt es, dazu die Bilder der Bausteine unter p zu berechnen und bei Octrees sind die einzelnen Teilwürfel abzubilden. Bei Polyederflächen bildet man jede Ecke und jede Kante einzeln ab, während bei einer parametrisierten Fläche F, gegeben durch $\boldsymbol{c} : [0,1]\times[0,1] \to \mathbb{R}^3$ man durch Verknüpfung mit p eine Abbildung [8] $\boldsymbol{c}' = p \circ \boldsymbol{c} : [0,1] \times [0,1] \to \mathbb{R}^2$ erhält, deren Bildmenge F' das Bild von F unter p ist. Um es wirklich auf dem Bildschirm zu erhalten, kann man z.B. über $[0,1] \times [0,1]$ ein hinreichend feines Raster legen und jedes Rechteck des Rasters mittels $\boldsymbol{c}'$ abbilden.

[8] $p \circ \boldsymbol{c}$ ist definiert durch $p \circ \boldsymbol{c}(\boldsymbol{x}) = p(\boldsymbol{c}(\boldsymbol{x}))$.

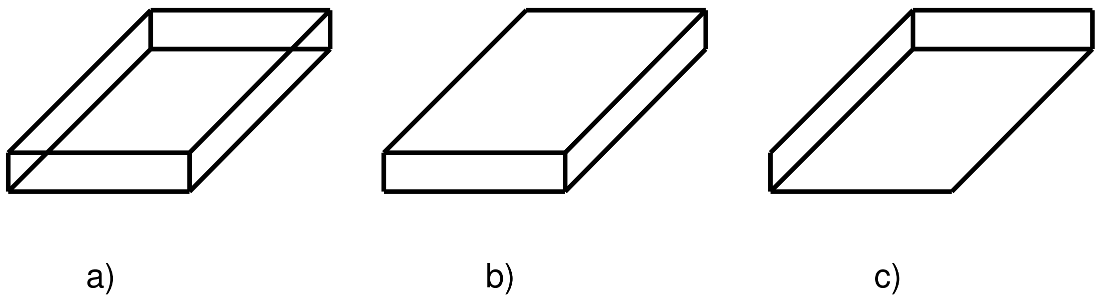

Abbildung 16: a) Drahtmodell eines Quaders; b) c) Interpretationsmöglichkeiten

3.2 Entfernung verdeckter Kanten und Flächen: Der Tiefenpufferalgorithmus

Für einfache Anwendungen der Computer-Graphik genügt es manchmal, insbesondere dann, wenn die auftretenden Objekte durch Polyederflächen dargestellt sind, alle Kanten gemäß einer im vorigen Abschnitt behandelten Projektion abzubilden. Bei diesem sogenannten *Drahtmodell* nimmt man also an, daß alle Teile des Objekte sichtbar, also nicht durch andere verdeckt sind. Für anspruchsvollere Bilder ist es jedoch notwendig, festzustellen, welche Teile von Kanten und Flächen verdeckt sind, und diese nicht erscheinen zu lassen (dieses wurde bei den Abbildungen dieses Abschnitts bereits getan). Abbildung 16a) zeigt das Drahtmodell eines Quaders, 16b) und c) zeigen zwei mögliche Interpretationen, je nachdem, welche Kanten man als verdeckt ansieht.
Das gebräuchlichste Verfahren zum Entfernen verdeckter Kanten und Flächen ist der sogenannte *Tiefenpufferalgorithmus*, der im folgenden vorgestellt werden soll. Wie im Abschnitt 3.1 soll angenommen werden, daß die x-y-Ebene des dreidimensionalen Raumes die Bildebene ist und das Projektionszentrum auf der negativen z-Achse liegt (bei Parallelprojektion im Unendlichen).
Beim Tiefenpufferalgorithmus wird angenommen, daß die Bildebene aus einzelnen Rasterpunkten besteht, was ja bei vielen Ausgabemedien (Bildschirm, Matrixdrucker) realistisch ist. Die Objekte der abzubildenden Szene sollen sich dadurch unterscheiden, daß sie verschiedene Farben oder Grauwerte haben. Der Algorithmus funktioniert nun so, daß er durch jeden Rasterpunkt P der Bildebene einen Sehstrahl vom Projektionszentrum Z aus (bei Parallelprojektion senkrecht zur Bildebene) legt und die Schnitte dieses Strahls mit den Objekten der Szene berechnet. Natürlich muß in P der Schnittpunkt (d.h. seine Farbe) abgebildet werden, der der Bildebene an nächsten ist, alle anderen sind verdeckt (siehe Abb.17) Dazu muß also der am wenigsten tiefliegende Schnittpunkt, das heißt der mit der kleinsten z-Koordinate bestimmt werden. Hat das Ausgabemedium m Rasterpunkte und die Szene n Objekte so müssen bis zu $n \cdot m$ Schnittpunkte getestet werden. Die Laufzeit des Algorithmus auf einem Computer ist also proportio-

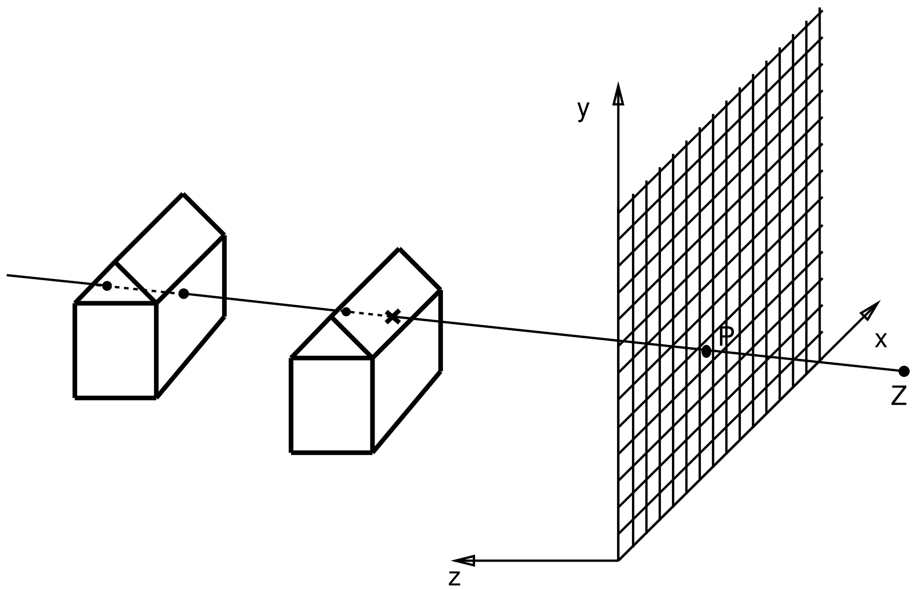

Abbildung 17: Ein Sehstrahl beim Tiefenpufferalgorithmus.
• - Schnittpunkte, × - abzubildender Punkt.

nal zu $n \cdot m$. Bei modernen hochauflösenden Bildschirmen liegt m etwa in der Größenordnung von einer Million, wie groß n ist, hängt von der Darstellung der dreidimensionalen Szene ab. Bei Darstellung durch CSG oder parametrisierte Oberflächen ist n relativ gering, bei Polyederflächen dagegen sehr hoch, da jedes einzelne Flächenelement betrachtet werden muß. In diesem Fall ist es sinnvoller, die Projektionen aller Flächenelemente auf den Bildschirm zu berechnen und für jeden Rasterpunkt P die Menge der darauf abgebildeten Flächenelemente einschließlich ihrer "Tiefe" (z-Koordinate) zu speichern. Am Ende wird dann jeweils das mit geringster Tiefe wirklich abgebildet.

4 Beleuchtung

Bei vielen Anwendungen der Computer-Graphik genügt es nicht, alle sichtbaren Teile der Szene auf den Bildschirm zu projizieren. Um das Bild "photorealistisch" zu machen, muß bei der Abbildung berücksichtigt werden, daß die Szene von einer oder mehreren Lichtquellen beleuchtet wird, daß Gegenstände Schatten werfen, und daß es matte, glänzende, durchsichtige oder gar spiegelnde Objekte in der Szene gibt (siehe Tafel 1)[9].

[9]Tafel 1 wurde nicht in die vorliegende Datei mit aufgenommen sondern befindet sich separat auf den Webseiten der Electronic Library, siehe http://www.emis.de/monographs/schulz/alt.html.

4.1 Schattierungen

Gegenstände, die nicht selbst leuchten, werden nur dadurch für das Auge sichtbar, daß sie auf sie fallendes Licht (teilweise) reflektieren. Dabei unterscheidet man zwischen *diffuser Reflexion*, bei der ein einfallender Lichtstrahl nach allen Seiten reflektiert wird, und *spiegelnder Reflexion*, bei der er nur in einer Richtung reflektiert wird, und zwar im gleichen Winkel zur Oberfläche, in dem er auftrifft. Die Reflexion der meisten Oberflächen ist eine Mischung aus beiden Reflexionsarten, wobei eine Oberfläche "matt" erscheint, wenn die diffuse und "glänzend", wenn die spiegelnde Reflexion überwiegt.
Ein relativ einfaches, in der Computer-Graphik benutztes Beleuchtungsmodell wurde von *Bui-Tuong Phong* (1975) entwickelt. Dabei wird angenommen, daß die in Richtung des Betrachters angegebene Lichtintensität I, die von einem Punkt einer Oberfläche ausgeht, sich zusammensetzt aus Reflexion des Lichtes einer oder mehrerer Lichtquellen nach beiden genannten Reflexionsarten, sowie aus Reflexion einer gewissen vorhandenen Hintergrundbeleuchtung ("ambient light"). Dieses "Streulicht" entsteht durch diffuse Reflexion an allen Gegenständen der Szene. Formal haben wir also

$$I = I_a + I_d + I_s$$

wobei I_a die durch Hintergrundbeleuchtung, I_d die durch diffuse und I_s die durch spiegelnde Reflexion entstandene Intensität ist. Die einzelnen Intensitäten berechnen sich wie folgt:

$$I_a = k_a \cdot I_H,$$

wobei I_H die Intensität des Hintergrundlichtes und k_a, $0 \leq k_a \leq 1$ der sogenannte *ambiente Reflexionskoeffizient* der Oberfläche ist.
Für die diffuse und spiegelnde Reflexion nehmen wir eine punktförmige Lichtquelle an. Bei mehreren solcher Lichtquellen muß über alle so erhaltenen Intensitäten summiert werden. Die Intensität durch diffuse Reflexion ist nach dem *Gesetz von Lambert* proportional zum Kosinus des Winkels zwischen dem Vektor $\boldsymbol{N}$, der am betrachteten Punkt senkrecht auf der Oberfläche steht (*der Oberflächennormalen*), und dem zur Lichtquelle hin zeigenden Vektor $\boldsymbol{L}$. Wir wählen $\boldsymbol{N}$ und $\boldsymbol{L}$ so, daß sie beide die Länge 1 haben, dann entspricht besagter Kosinus ihrem Skalarprodukt und wir erhalten

$$I_d = I_Q \cdot (\boldsymbol{N} \cdot \boldsymbol{L}) \cdot k_d$$

wobei I_Q die Intensität der Lichtquelle, und k_d, $0 \leq k_d \leq 1$ der von der Oberfläche und Wellenlänge des Lichts abhängige *diffuse Reflexionskoeffizient* ist. Für die spiegelnde Reflexion liefert folgender Ansatz gute Ergebnisse:

$$I_s = k_s \cdot I_Q \cdot (\boldsymbol{A} \cdot \boldsymbol{R})^c$$

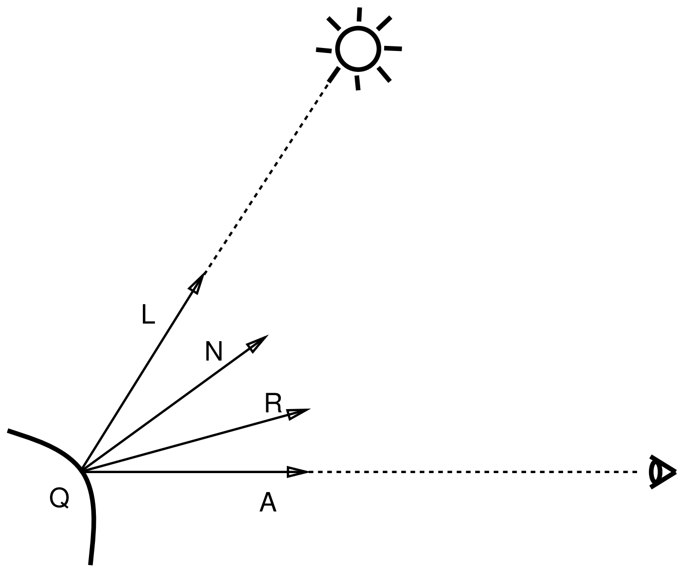

Abbildung 18: Lichteinfall und Reflexion an einem Oberflächenpunkt.

wobei k_s, $0 \leq k_s \leq 1$, der spiegelnde Reflexionskoeffizient, $\boldsymbol{A}$ der Vektor vom betrachtenden Oberflächenpunkt zum Augpunkt und $\boldsymbol{R}$ der Vektor in Richtung des reflektierenden Lichtstrahls ist. Beide Vektoren haben Länge 1 (siehe Abb. 18). c ist der sogenannte *Materialkoeffizient*, hängt also vom Material des beleuchtenden Gegenstands ab. Er liegt meist zwischen 1 und 100; je größer c ist, umso kleiner und konzentrierter ist das Spiegelbild der Lichtquelle auf der Oberfläche. Mit diesen Formeln lassen sich nun die Lichtintensitäten jedes sichtbaren Punktes einer Szene berechnen und man erhält damit ein entsprechend schattiertes Bild auf dem Bildschirm. Ein dabei noch auftretendes Problem ist das von *Schatten*, also Bereichen, die von der Lichtquelle aus wegen dazwischenliegender Objekte nicht durch einen direkten Strahl erreicht werden. Diese Bereiche reflektieren also lediglich das Hintergrundlicht nicht die Lichtquelle selbst. Da sie genau die von der Lichtquelle aus gesehen *verdeckten* Bereiche sind, lassen sie sich mit den Methoden aus Abschnitt 3.2 bestimmen.
Die hier angegebenen Formeln sind, wie gesagt, nur ein grobes Modell der Schattierung von Oberflächen. Trotzdem lassen sich schon relativ realistische Effekte erzeugen (siehe Tafel 2)[10].
Bei einem weiter verfeinerten Modell würde man z.B. auch die Entfernung der Lichtquelle von den einzelnen Gegenständen berücksichtigen. Im Fall, daß durchsichtige Gegenstände auftreten (siehe Tafel 1)[11] muß auch durchfallendes Licht

[10] Tafeln 1 und 2 wurden nicht in die vorliegende Datei mit aufgenommen, sondern befinden sich separat auf den Webseiten der Electronic Library, siehe http://www.emis.de/monographs/schulz/alt.html.
[11] Siehe die vorhergehende Fußnote.

unter Berücksichtigung der physikalischen Brechungsgesetze in obige Formeln mit eingebracht werden.
Will man statt Schwarzweiß-Bildern farbige erzeugen, so sind alle diese Berechnungen für z.B. drei Grundfarben (etwa Rot, Gelb und Blau) durchzuführen. Die Verschiedenheit der Reflexionskoeffizienten für die verschiedenen Grundfarben liefert dann die *Farbe* einer Oberfläche.

4.2 Strahlverfolgung

Die Strahlverfolgungsmethode ("ray tracing") kombiniert die bisher vorgestellten Ideen, um möglichst realistische Bilder zu erzeugen. Wie in Abschnitt 3.2.1 wird angenommen, daß die Bildebene ein quadratisches Raster ist. Vom Augpunkt Z aus wird nun ein Strahl s durch jeden Rasterpunkt P gelegt und (mit den Methoden von 3.2) das Objekt O bestimmt, das von s zuerst getroffen wird. (Gibt es kein solches Objekt, so erhält P die Hintergrundintensität.) Q sei der Auftreffpunkt von s auf O. Nun gibt es mehrere Möglichkeiten:

1. Ist O *matt und undurchsichtig*, so wird die Lichtintensität in Q durch die in 4.1 beschriebenen Methoden bestimmt und P zugeordnet.

2. Ist O *spiegelnd*, so wird der Strahl s an der Stelle Q reflektiert und weiterverfolgt. (Trifft er dann z.B. im Punkt Q' auf ein mattes Objekt O' so wird P die Intensität von Q', möglicherweise wegen der Spiegelung abgeschwächt, zugeordnet: O spiegelt sich in O').

3. Ist O *durchsichtig*, so wird s von Q aus gemäß der physikalischen Brechungsgesetze weitergeleitet.

Natürlich können auch die Fälle 2 und 3, das heißt Spiegelungen und Brechungen gleichzeitig auftreten (siehe die Oberfläche der Gläser in Tafel 1)[12]. In diesem Fall wird der Strahl s aufgespalten in einem reflektierenden Strahl s' und einem gebrochenen Strahl s''. Die Strahlverfolgung endet, wenn ein Strahl entweder auf einer matten, undurchsichtigen Oberfläche, im Hintergrund oder bei einer Lichtquelle ankommt.
Tafel 1 wurde mit Hilfe des Strahlverfolgungsverfahrens hergestellt[13]. Sie zeigt, von welch hoher Qualität damit erzeugte Bilder sein können. Leider sind, wie man sich aufgrund der Darstellungen in den letzten Abschnitten vorstellen kann, diese Verfahren sehr rechenintensiv. Besonders die Herstellung eines *Films*, bei der Tausende von Bildern erzeugt werden müssen, erfordert enorme Rechenzeiten, i.e. mehrere Stunden pro Minute Film.

[12]Tafel 1 wurde nicht in die vorliegende Datei mit aufgenommen sondern befindet sich separat auf den Webseiten der Electronic Library, siehe http://www.emis.de/monographs/schulz/alt.html.

[13]Siehe die vorhergehende Fußnote.

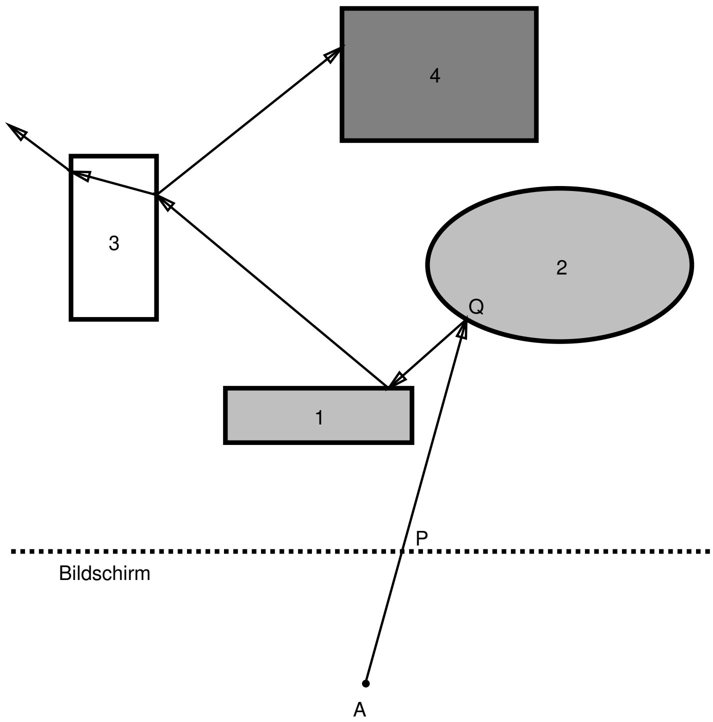

Abbildung 19: Strahlverfolgung: 1,2 spiegelnde; 3 spiegelndes und durchsichtiges; 4 mattes und undurchsichtiges Objekt.

Das Entwickeln effizienterer Methoden besonders bei Objekten mit regelmäßig geformten Oberflächen, wie sie etwa bei der CSG (Abschnitt 2.1) auftreten, ist unter anderen aktueller Forschungsgegenstand der Computer-Graphik.

Weiterführende Literatur

W.D. Fellner, *Computer-Grafik*, BI-Wissenschafts-Verlag, 1988.

J.D. Foley, A. van Dam, S.K. Feiner, J.F. Hughes, *Computer Graphics: Principles and Practice*, 2. Ausgabe, Addison-Wesley Publ. Comp., 1990.