Vorlesung ComputerGraphik

SS 1993

Arno Formella

Inhaltsverzeichnis

1 Einführung

1.1 Was ich NICHT mache

- Hardware: Graphikkarten, Displaysysteme, Graphikprozessoren,
- Hardwareeinflüsse auf die Software: Farbdarstellungen, Bildschirmauflösung, Hardware–Implmentation von Graphikalgorithmen
- 2D–ComputerGraphik
  - Darstellung 2dimensionaler Grundobjekte wie Linien, Polygone, Kreise, Ellipsen,
  - Elimination von verborgenen Kanten und verdeckten Oberflächen
- grundlegendes Handwerkszeug in der 2D/3D–Graphik
  - mathematische Grundlagen: Vektoralgebra, Koordinatensysteme, Skalieren, Translationen, Rotationen, Projektionen
  - Clipping
- Darstellungsmethoden:
  - Drahtgitterdarstellung: ohne oder mit verdeckten Kanten
  - Gouraud–Schattierung
  - Phong–Schattierung
  - Radiosity–Verfahren

  und deren speziellen Algorithmen: Scan–Line–Algorithmen, Z–Buffer–Algorithmen
- Bildnachbearbeitung: Anti–Aliasing, Dithering, Farbmodelle (?)

Notwendige Einzelheiten werden gegebenenfalls, oder auf Anfrage, entsprechend kurz erläutert; gewisse Erfahrungen auf diesem Gebiet helfen, sind jedoch nicht unbedingt Voraussetzung, da die behandelten Themen einen besonderen Zweig der ComputerGraphik abdecken, der nicht in den low level Bereich der ComputerGraphik fällt

Literatur:

- RAUBER, THOMAS; Vorlesungsskript: ComputerGraphik WS90/91; *Universiät des Saarlandes* demnächst auch als Buch, das auch einige der gemachten Themen abdeckt

- VAN DAM, FOLEY; Fundamentals of Interactive Computer Graphics; *Add. Wesley, 1982*
- Graphic Gems: Vol I, II und III viele der Algorithmen sind auch über ftp erhältlich
- HOSCHEK, LASSER; Grundlagen der geometrischen Datenverarbeitung; *Teubner, 1989*

1.2 Was ich mache

Realistische (fotorealistische) Aspekte der ComputerGraphik

Ziel ist es realitätsnahe Darstellungen von Objekten zu erzeugen. Hierzu müssen die optischen und geometrischen Eigenschaften der Objekte als auch die der Lichtquellen modelliert werden. Dies kann nur näherungsweise erfolgen, oft sogar nur nach dem Prinzip: Das mithilfe des Modell erzeugte Bild sieht — mit etwas Wohlwollen — realistisch wie eine Fotographie aus.

Beleuchtungsmodelle Es werden ausgehend von den physikalischen Grundlagen über die in der ComputerGraphik gemachte Vereinfachungen alle wesentlichen Beleuchtungsmodelle vorgestellt und motiviert. Hierbei wird allerdings Radiosity ausgeklammert.

RayTracing Es wird der Grundalgorithmus vorgestellt. Dabei werden besonders die Raumaufteilungsverfahren erläutert. Die Integrationsmöglichkeiten obiger Beleuchtungsmodelle wird diskutiert. Beschleunigungsverfahren, insbesondere Caching–Strategien und einfache Parallelisierungen, werden analysiert.

gekrümmte Linien und Oberflächen Es werden die Grundlagen zur Darstellung gekrümmter Oberflächen gelegt; Stichworte sind BEZIÉR–Flächen, Splines und NURBS.

Texturen und Oberflächenstrukturen Sollte noch Zeit bleiben, werden die Erzeugungsalgorithmen gängiger dreidimensionaler Texturen vorgestellt.

Literatur wird in den entsprechenden Kapiteln angegeben

Übungen Die Übungen verfolgen im wesentlichen vier Ziele: Nachrechnen von mathematischen Kleinigkeiten, Motivation zur eigenen Modellierung erwünschter Effekte, Aufbau einer Sammlung von Modellierungstechniken (und somit render–Techniken), Äußerung des Wunsches, sich mit der Materie zu befassen.

2 Beleuchtungsmodelle

Literatur:

- RAUBER, THOMAS; Algorithmen in der Computer Graphik; *erscheint in Springer Verlag, voraus. 1993*
- FOLEY, JAMES; VAN DAM, ANDRIES; Computer Graphics, Principles and Practice; *2nd Edition, Addison Wesley, 1991*
- CLAUSSEN, UTE; Beleuchtungsmodelle und Beleuchtungsalogrithmen in der Graphischen Datenverarbeitung; *Bericht WSI–GRIS 88–3, Universität Tübingen, 1988*
- HALL, ROY; Illumination and Color in Computer Generated Imagery; *Springer Verlag, New York, 1989*
- SIEGEL, ROBERT; Thermal Radiation Heat Transfer; *2. Edition, Hemisphere Publishing Corporation, 1981*
- HE, XIAO: TORRANCE, KENNETH; SILLION, FRANCOIS; GREENBERG, DONALD; A Comprehensive Physical Model for Light Reflection; *Computer Graphics, Vol. 25, No. 4, 1991*

Ein Beleuchtungsmodell beschreibt:

- physikalische Eigenschaften des einfallenden Lichtes
  - Farbverteilung
  - Helligkeit, Intensität
  - Geometrie der Lichtquelle
- physikalische Eigenschaften des Objektes
  - Oberflächengeometrie
  - optische Eigenschaften des Objektes, Materialeigenschaften: Reflexions–, Emission–, Absorptions–, Transmissions–Eigenschaften
- wie aus den Eigenschaften der Lichtquelle und den Objekten die Intensitäten an den Oberflächen der Objekte berechnet werden können, so daß diese mit irgendwelchen Methoden auf dem Ausgabemedium dargestellt werden können.

2.1 Physikalische Grundlagen

Man muß zwischen Modell, auch physikalischem Modell, und Gesetz unterscheiden: Modelle sind Erklärungsversuche, z.B. Thesen vor der Ausführung eines physikalischen Experimentes, Gesetze sind Beschreibungen der Realität, die durch Experimente geprüft worden sind und von denen man behaupten kann, daß sie in der Realität gelten; zumindest gibt es keinen Gegenbeweis. Noch nicht in allen Einzelheiten überprüfte Modelle bezeichnet man auch als Theorien. Modelle gelten meist nur in gewissen Bereichen. Auch Gesetze erlauben Näherungen (z.B. klassische Mechanik und Quantenmechanik).

In der ComputerGraphik genügen meist Modelle.

2.1.1 Einige Bemerkungen zur Modellierung des Lichtes

physikalische Modellierung: objektive, meßbare Eigenschaften des Lichtes im Sinne der Physik,

physiologische, technische Modellierung: subjektive, individuelle Eigenschaften der Wahrnehmung im Sinne der Physiologie (auch Psychologie)

Wellenmodell der Physik beschreibt Licht als elektromagnetische Welle (Maxwellschen Gleichungen),

Teilchenmodell der Physik beschreibt Licht als masselose Photonen

Quantenmechanik vereinheitlicht beide Modelle

Strahlenoptik oder geometrische Optik betrachtet Licht nur als Strahlen in Richtung des Energietransportes

Je nach Experiment und Modellierung nimmt man die eine oder die andere Beschreibungsmethode, erklärbare Effekte:

Wellenmodell Beugung, Dispersion, Streuung, Interferenz, Polarisation ...

Teilchenmodell Photoeffekt, Compton–Effekt, Emission, Absorption ...

Strahlenoptik Brechung, Reflexion, ...

Wir wollen nicht — obigem Prinzip folgend — die Physik bis ins letzte Detail der Quatenmechanik simulieren, sondern machen plausible Vereinfachungen je nach Darstellungsmodell (und es war gerade die Leistung, z.B. von PHONG, COOK, TORRANCE, BLINN ua., gerade solche Vereinfachungen zu wählen, daß sich das ergebende Modell zur Berechnung von realitätsnahen Bildern eignet)

Folgende generelle Annahmen über das Licht werden gemacht:

- Licht breitet sich geradlinig aus, d.h. Beschränkung im wesentlichen auf die Strahlenoptik
- es gilt die Superpositionseigenschaft des Lichtes, d.h. die Einflüsse mehrerer Lichtquellen verhalten sich additiv
- makroskopischer Sichtweise: Linearkombination von idealer und diffuser Reflexion (Remission) sowie von idealer und diffuser Transparenz (Transluzenz)

Bemerkung: Sollen nicht realistische Effekte in der ComputerGraphik genutzt werden, kann natürlich von diesen Annahmen abgewichen werden.

2.1.2 Strahlungslehre

Notationen übliche Vektorraum–Mathematik

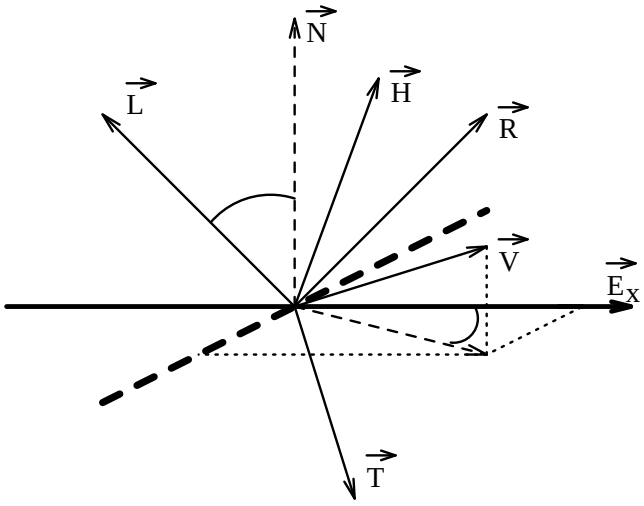

Abbildung: Vektornotationen

$\vec{E}$	Vektoren eines Koordinatensystems
$\vec{H}$	Halbvektor zwischen $\vec{L}$ und $\vec{V}$
$\vec{L}$	Vektor in Richtung einer Lichtquelle
$\vec{N}$	Normalenvektor einer Ebene
$\vec{R}$	Reflexionsvektor an einer Ebene
$\vec{T}$	Transmissionsvektor an einer Ebene
$\vec{V}$	Vektor in Richtung des Betrachters (view)
δ	Polar–,Divergenzwinkel (zwischen $\vec{X}$ und $\vec{N}$)
φ	Azimut, Rotationswinkel (zwischen Vektorebene und 1. Koordinatenachse)

Es gilt je nach betrachtetem Vektor $\vec{X}$

$$\cos\delta = \langle \vec{X}, \vec{N} \rangle = \frac{\vec{X} \cdot \vec{N}}{|\vec{X}|\,|\vec{N}|}$$

$$\cos\varphi = \langle \vec{X} \times \vec{N}, \vec{E_x} \rangle$$

Raumwinkel

$$\Omega = A/r^2$$

wobei A eine Teiloberfläche auf einer Kugel mit Radius r ist, die durch den betrachteten Winkels aufgespannt wird ($4\pi = 4\pi r^2/r^2$ entspricht der gesamten Kugel) entspricht Bogenmaß im 2dimensionalen

Um den Raumwinkel unabhängig von einem Radius r zu machen, betrachten wir ein durch δ und φ aufgespanntes Koordinatensystem, d.h. die Menge

$$\{(\delta, \varphi) | \delta \in [0, \frac{\pi}{2}] \text{ und } \varphi \in [0, 2\pi]\}$$

beschreibt eine Halbkugel, oder einen Raumwinkel $\Omega = 2\pi$.

Für einen differentiellen Raumwinkel $d\Omega$ gilt dann:

$$d\Omega = \sin\delta \; d\delta d\varphi$$

was man durch folgende Abbildung leicht einsieht

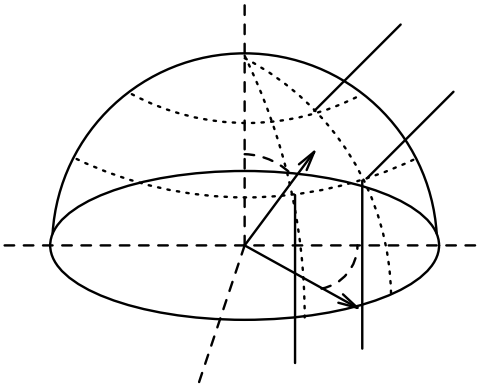

Abbildung: differentieller Raumwinkel

Den Raumwinkel einer Halbkugel erhält man damit als

$$\begin{aligned}
\Omega &= \int_H d\Omega \\
&= \int_{\varphi=0}^{2\pi} \int_{\delta=0}^{\frac{\pi}{2}} \sin\delta \; d\delta d\varphi \\
&= \int_{\varphi=0}^{2\pi} [-\cos\delta]_0^{\frac{\pi}{2}} \; d\varphi \\
&= 2\pi(0-(-1)) \\
&= 2\pi
\end{aligned}$$

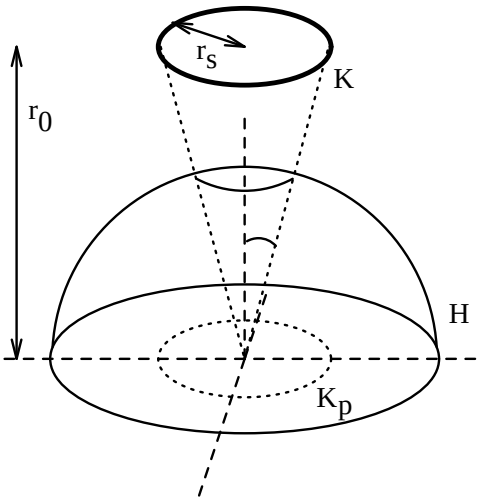

Abbildung: Raumwinkelberechnung einer Kreisscheibe

Integriert man nicht über den gesamten Halbraum bezüglich δ, d.h. man betrachtet nur einen Teilwinkel, der sich z.B. als Projektion einer Kreisscheibe K auf die Hemisphere ergibt (siehe Abbildung), erhält man

$$\begin{aligned}
\Omega_K &= \int_{\varphi=0}^{2\pi} \int_{\delta=0}^{\delta_K} \sin\delta \; d\delta d\varphi \\
&= 2\pi(1-\cos\delta_K)
\end{aligned}$$

Betrachte Kreisscheibe K mit Radius r_s im Abstand r_0 vom Beobachtungspunkt; die Projektion auf die Einheitskugel liefert Polarwinkel δ

$$\delta = \arctan(\frac{r_s}{r_0})$$

somit

$$\Omega_K = 2\pi(1 - \cos(\arctan(\frac{r_s}{r_0})))$$

was auch der Formel

$$A = 2\pi h$$

als Oberfläche eines Einheitskugelsegments der Höhe h entspricht.

Liegt nun die Kreisscheibe weit vom Beobachtungspunkt entfernt, d.h. r_0 ist wesentlich größer als r_s kann man die Näherung

$$\delta \approx \sin\delta \approx \arctan(\frac{r_s}{r_0}) \approx \frac{r_s}{r_0}$$

verwenden und man erhält:

$$\begin{aligned} \Omega_K &\approx \int_{\varphi=0}^{2\pi} \int_{\delta=0}^{\frac{r_s}{r_0}} \delta \, d\delta d\varphi \\ &= \pi \frac{r_s^2}{r_0^2} \end{aligned}$$

Nun ist aber πr_s^2 gerade die Fläche der betrachteten Kreisscheibe, d.h. man kann kleine Raumwinkel, die sich als Projektion einer Fläche A ergeben, durch die Projektionsfläche in der Betrachterebene dividiert durch den quadratischen Abstand annähern:

$$d\Omega \approx \frac{\langle \vec{W}, \vec{N}(A) \rangle \; A}{r_0^2}$$

Übungen

1. Verifizieren sie die Formel $A = 2\pi h$.
2. Berechnen Sie den Raumwinkel eines beliebigen Polygons P, wobei Betrachtervektor $\vec{V}$ und Normalenvektor $\vec{N}(P)$ kolinear sind.
3. Berechnen Sie den Raumwinkel eines beliebigen Polygons P, wobei Betrachtervektor $\vec{V}$ und Normalenvektor $\vec{N}(P)$ nicht kolinear sind.
4. Berechnen Sie den exakten Raumwinkel einer Kreisscheibe, wenn Betrachtervektor $\vec{V}$ und Normalenvektor $\vec{N}(K)$ nicht kolinear sind.
5. Berechnen Sie den Fehler in Abhängigkeit vom Verhältnis r_s/r_0, der durch obige Approximation des Raumwinkels gemacht wird.

Strahlungsarten Wir betrachten die Eigenschaften der Oberflächen zunächst unabhängig von der Wellenlänge. Dies wird in späteren Abschnitten geändert.

Man kann folgende Strahlungsarten mit den entsprechenden Koeffizienten unterscheiden, dabei bedeutet ein dicker Pfeil gerichtete Strahlung und dünne Pfeile diffuse Strahlung.

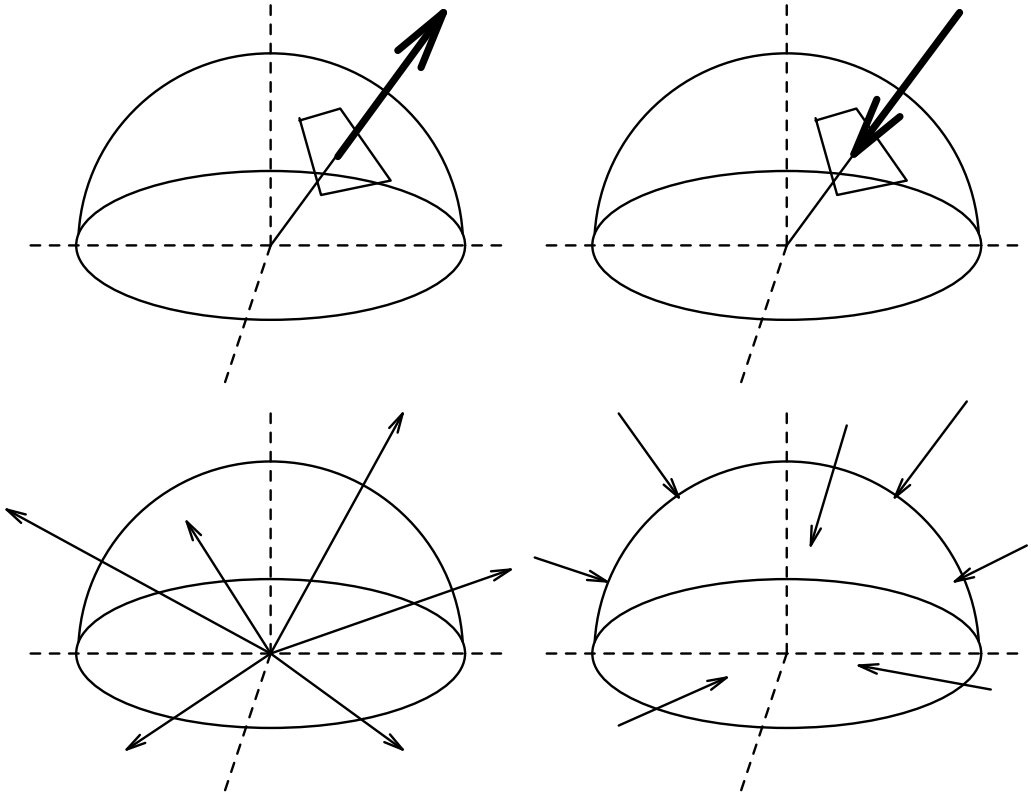

Abbildung: Strahlungsarten, Teil 1

gerichtete Emmisionsrate *directional emissivity*	ϵ_s
diffuse Emmisionsrate *hemisperical emissivity*	ϵ_d
gerichtete Absorptionsrate *directional absorptivity*	α_s
diffuse Absorptionsrate *hemisperical absorptivity*	α_d

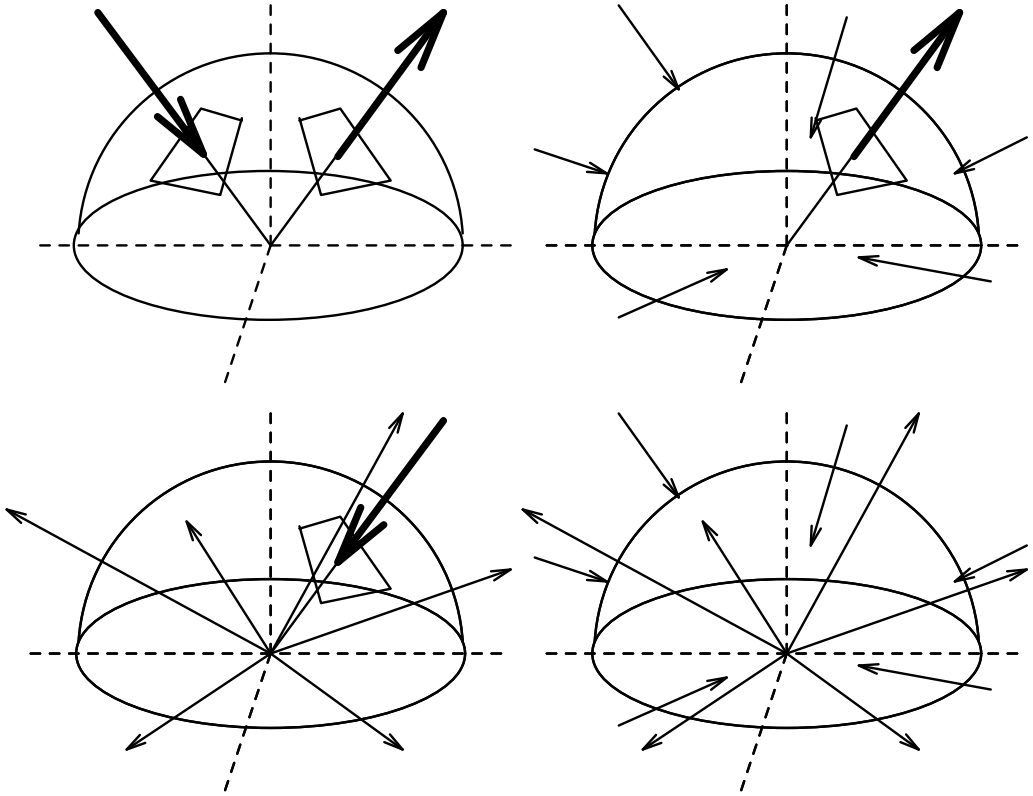

Abbildung: Strahlungsarten, Teil 2

gerichtete Reflexionsrate bei gerichteter Immision *bidirectional spectral reflectivity*	ρ_s
gerichtete Reflexionsrate bei diffuser Immision *directional–hemispherical reflectivity*	ρ_d
diffuse Reflexionsrate bei gerichteter Immision *hemispherical–directional reflectivity*	
diffuse Reflexionsrate bei diffuser Immision *hemispherical reflectivity*	

Wie die einzelnen Koeffizienten berechnet werden und welche in der ComputerGraphik eine Rolle spielen, sehen wir in den folgenden Ausführungen.

Definitionen

- Der *Strahlungsfluß* Φ [J/s] oder auch die Strahlungsleistung wird von einer Strahlungsquelle in den gesamten Raum abgegeben (Teilchenmodell: Anzahl der pro Zeiteinheit abgegebenen Photonen).
- Die *Strahlungsmenge* W [J] ist die in einem Zeitintervall von einer Strahlungsquelle abgegebene Energie.

$$W = \int_{t_0}^{t_1} \Phi \; dt$$

- Die *Strahlungsstärke* J [J/s] ist die in einen Raumwinkel abgegebene Strahlungsleistung

$$J = \frac{d\Phi}{d\Omega}$$

Man erhält somit die Strahlungsleistung als

$$\Phi = \int_{\Omega} J \; d\Omega$$

Strahler Die Strahlungsstärke ist eine Eigenschaft des Strahlers und unabhängig von der Entfernung eines möglichen Beobachters. Ist $dJ/d\Omega$ konstant, handelt es sich um einen isotropen Strahler, der in alle Richtungen die gleiche Strahlungsleistung abgibt. Es gilt dann:

$$\Phi = J\ \Omega$$

also für den gesamten Raum

$$\Phi = 4\pi\ J$$

Isotrope Strahler kommen jedoch kaum vor, da Strahler eine Geometrie besitzen und jedes Flächenstück dS des Strahlers einen Teil zur Gesamtstrahlungsleistung Φ beiträgt (in großen Entfernungen können jedoch auch nicht–isotrope Strahler als isotrope Strahler modelliert werden, z.B. Sonnenlicht oder vereinfachte Punktlichtquellen)

Gilt nun, daß die Strahlungsstärke eines Flächenstücks dS nur von der Beobachtungsrichtung $\cos\delta = \langle \vec{W}, \vec{N}(S) \rangle$ gemäß

$$J(\delta) = B\ dS\ \cos\delta$$

abhängt, wobei B eine Konstante, nämlich die *Strahldichte* des Flächenstücks, ist, dann nennt man den Strahler auch LAMBERT–Strahler.

Die Strahlungsleistung des gesamten Flächenstücks (in einen Halbraum) beträgt dann

$$\begin{aligned}
\Phi &= \int_H J(\delta)\ d\Omega \\
&= \int_H B\ dS \cos\delta\ d\Omega \\
&= B\ dS \int_{\varphi=0}^{2\pi} \int_{\delta=0}^{\pi/2} \cos\delta \sin\delta\ d\delta\ d\varphi \\
&= 2\pi B\ dS \int_0^{\pi/2} \cos\delta \sin\delta\ d\delta \\
&= 2\pi B\ dS \left[\frac{1}{2}\sin^2\delta\right]_0^{\pi/2} \\
&= \pi\ dS\ B
\end{aligned}$$

Bestrahlte Fläche Herrscht auf einem Flächenstück dE der Strahlungsfluß $d\Phi$, bezeichnet D die *Intensität* (Strahlungsflußdichte) auf dem Flächenstück mit

$$D = \frac{d\Phi}{dE}$$

Dies ist letztendlich das, was in der ComputerGraphik dargestellt werden soll. Dabei wird meistens der Abstand zwischen der Projektionsebene und dem Objekt nicht mehr berücksichtigt, d.h. auf dem Bildschirm wird das Objekt mit der Intensität dargestellt, die das Beleuchtungsmodell auf dessen Oberfläche berechnet. Dies kann jedoch aufgegeben werden.

Oder anders: Den Gesamtfluß, der auf einer Fläche herrscht, erhält man als Oberflächenintegral

$$\Phi = \int_A \vec{D} d\vec{E}$$

Spätestens hier erkennt man, daß man eigentlich mit gerichteten Größen operieren muß, was wir aus Vereinfachungsgründen unterlassen haben.

Analog kann man die spezifische Ausstrahlung R (oder Intensität des Strahlers) definieren, die die von einem Oberflächenstück dS abgegebene Strahlungsleistung spezifiziert.

$$R = d\Phi/dS$$

Nochmal:

- Intensitäten sind Flußdichten auf einer Fläche
- Strahlungsstärken sind Flußdichten in einem Raumwinkel
- Strahldichten sind Strahlungsflüsse pro Raumwinkel und Fläche

Interaktion: Strahler und bestrahlte Fläche Betrachten wir einen Strahler, der ein Flächenstück dE aus einem Raumwinkel Ω von hinreichend weiter Entfernung beleuchtet. Dieses erscheint dem Strahler als Raumwinkel

$$d\Omega_E = \frac{dE}{r_0^2} \cos \delta_E$$

wobei r_0 der Abstand zwischen Strahler und Fläche ist und $\delta_E = \langle \vec{L}, \vec{N}(E) \rangle$ der Verkürzungsfaktor.

Nehmen wir nun an, daß die Strahlungsstärke $J(dS)$ des Strahlers in dem Bereich annähernd konstant ist, folgt für den Strahlungsfluß bezüglich des Flächenstücks dE

$$\begin{aligned} d\Phi &= J(dS) d\Omega_E \\ &= J(dS) \frac{dE}{r_0^2} \cos \delta_E \end{aligned}$$

Handelt es sich um einen LAMBERT–Strahler, gilt weiterhin:

$$J(dS) = B dS \cos \delta_S$$

wobei $\delta_S = \langle \vec{W}, \vec{N}(S) \rangle$ die Beobachtungsrichtung der Lichtquelle ist.

Wir erhalten also für den Strahlungsfluß der Fläche dE

$$d\Phi = B \frac{dS dE}{r_0^2} \cos \delta_S \cos \delta_E$$

oder für die Intensität

$$\begin{aligned} D &= \frac{d\Phi}{dE} \\ &= B \frac{dS}{r_0^2} \cos \delta_S \cos \delta_E \\ &= B d\Omega_S \cos \delta_E \end{aligned}$$

d.h. die Intensität folgt ebenfalls dem LAMBERTschen Cosinus–Gesetz, jetzt bezüglich des Winkels zwischen Lichtquelle und Oberfläche. Dabei wird mit einem LAMBERT–Strahler aus einem Raumwinkel $d\Omega_S$ bestrahlt, der eine Strahldichte B hat.

Fassen wir nochmals die Bedingungen für die Näherungen zusammen:

- Das Verhältnis der Fläche des Strahlers zum Abstand, sowie das Verhältnis der bestrahlten Fläche zum Abstand sind klein, so daß $\delta \approx \sin \delta$ und somit $d\Omega \approx dA_p \cos \delta$ gilt.
- Der Strahler ist ein LAMBERT–Strahler und folgt dem Cosinus–Gesetz $J(\delta) = B dS \cos \delta$.
- Der Strahlungsfluß ist auf der bestrahlten Fläche konstant.

Obige Gleichungen sind eigentlich auch von der Wellenlänge abhängig, d.h. vorallem die Intensität ist mit der Wellenlänge korrelliert. Wir werden die Abhängigkeit nur in bestimmten Fällen berücksichtigen.

2.1.3 Physikalische Größen und Gesetze

Größen

- Lichtgeschwindigkeit c, v

  im Vakuum $c = 2.998 * 10^8$ m/s
  in Material $v = c/n = \sqrt{\mu\varepsilon}$

  n Brechzahl
  ε Dielektrizitätszahl
  μ Permeabilitätszahl

  $\mu \approx 1$ für die meisten transparenten Materialien
  $\varepsilon = [1..100]$

- Frequenz ν

$$\nu = E/h$$

  E Energie
  h PLANCKsche Konstante ($= 6.626 \ \ 10^{-34}$ Js)

- Wellenlänge λ

$$\lambda = v/\nu$$

  Wahrnehmung des Menschen ca. 380 bis 770 nm Wellenlänge siehe Verteilung in Abbildung

Gesetze Die folgenden Gesetze werden ohne Herleitung präsentiert. Es sei gesagt, daß sie aus den MAXWELLschen Gleichungen und einigen Nebenbedingungen (Energieerhaltungssatz u.a.) hergeleitet werden können (siehe z.B. [Siegel]). Die ersten beiden Gestze zeigen, wie die Strahldichte bzw. die spezifische Ausstrahlung idealerweise durch Modelle der Physik berechnet werden können. Dies ist jedoch in der ComputerGraphik nicht ausreichend, da es einerseits nicht um die Darstellung idealisierter Vorgänge geht und andererseits die Strahlungsquellen durch die Angabe der Intensitäten bzw. Strahldichten modelliert werden, d.h. eine Modellierung aufgrund anderer physikalischer Eigenschaften ist nicht notwendig. Sie sind hier angeführt, um den Zusammenhang zwischen Temperatur und Farbe (Wellenlänge) der Lichtquelle herstellen zu können, sowie die Intensitätsdämpfung einer Lichtquelle aufgrund des umgebenden Mediums. Beide Effekte können u.U. in die Modellierung der ComputerGraphik eingebunden werden.

PLANCK**sches Gesetz** Das PLANCKsche Gesetz gibt an, welche Strahldichte B_λ ein schwarzer Körper mit einer bestimmten Wellenlänge λ bei einer Temperatur T aufweist:

$$\begin{aligned} B_\lambda &= \frac{2C_1}{\lambda^5(e^{\frac{C_2}{\lambda T}} - 1)} \\ C_1 &= hc \\ C_2 &= \frac{hc}{k} \end{aligned}$$

wobei h die PLANCKsche Konstante, c die Lichtgeschwindigkeit und k die BOLTZMANN Konstante ($1.381 * 10^{-23}$ J/K) ist. Integriert man über die Wellenlänge, erhält man:

$$\begin{aligned} B &= \int_0^\infty B_\lambda \, d\lambda \\ &= \frac{2C_1\pi^5}{15C_2^4}\frac{1}{\pi}T^4 \\ &= \frac{\sigma}{\pi}T^4 \end{aligned}$$

Ist der umgebende Raum nicht das Vakuum, muß c durch v als Lichtgeschwindigkeit im Medium ersetz werden.

Damit ergibt sich als spezifische Ausstrahlung eines schwarzen Körpers mit der Oberfläche S über den gesamten Wellenlängenbereich:

$$\begin{aligned} R &= \frac{\Phi}{S} \\ &= \frac{\pi B S}{S} \\ &= \sigma T^4 \end{aligned}$$

(STEFAN–BOLTZMANN Gesetz). Hier ergibt sich bei umgebendem Medium mit Brechungsindex n

$$R = n^2 \sigma T^4$$

LAMBERT**sches Gesetz** Das LAMBERTsche Gesetz wurde schon bei der Interaktion zwischen Strahler und bestrahlter Fläche erläutert. Es besagt, das die Intensität bzw. die Strahlungsstärke nur vom Cosinus des Betrachtungswinkels abhängt.

SNELL**sches Gesetz** Es gilt einerseits das Reflexionsgesetz:

$$\delta_l = \delta_r \qquad (= \delta)$$

sowie das SNELLsche Gesetz:

$$\frac{\sin \tau}{\sin \delta} = \frac{n_1}{n_2}$$

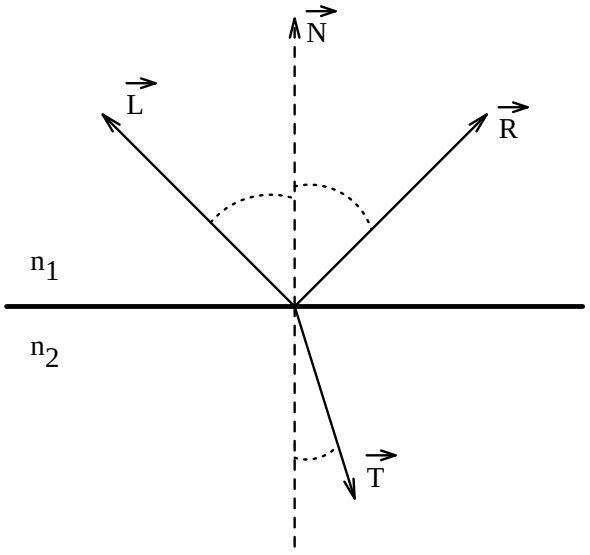

Abbildung: SNELLsches Gesetz

Die Brechungsindizes sind von der Wellenlänge abhängig, d.h. man muß das Gesetz mit $n(\lambda)$ formulieren. Sind die Materialien nicht isotrop bezüglich der elektromagnetischen Eigenschaften, ist der Brechungsindex sogar komplex, d.h. abhängig von ϕ. Auf diesen Fall wird bei den FRESNELschen Gleichungen kurz eingegangen.

Ab einem gewissen Eintrittswinkel tritt Totalreflexion auf, d.h. der Strahl aus dem dichteren Medium (größerer Brechungsindex) kann nicht mehr in das dünnere Medium eindringen.

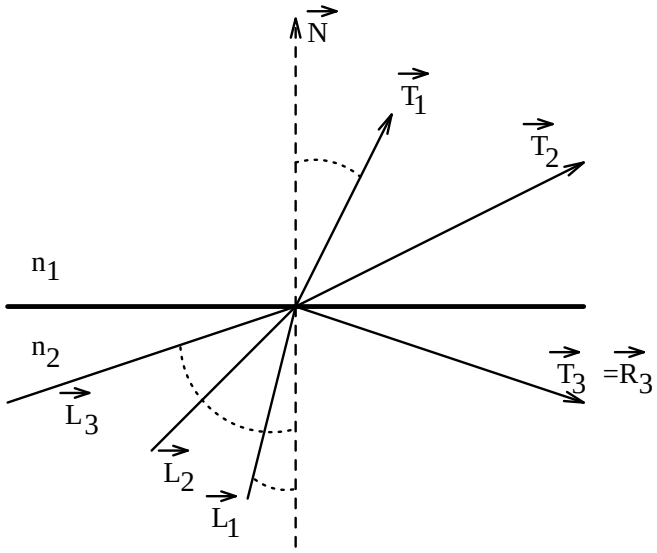

Abbildung: Totalreflexion

Dies tritt genau dann ein, wenn für den Eintrittswinkel δ gilt:

$$\sin\delta \geq \sin\delta_g = \frac{n_1}{n_2} \qquad (n_2 > n_1)$$

FRESNEL**sche Gleichung**

$$\begin{aligned}
\rho_d(\delta) &= \frac{1}{2}(\rho_{\|} + \rho_{\perp}) \\
&= \frac{1}{2}\left(\frac{\tan^2(\delta-\tau)}{\tan^2(\delta+\tau)} + \frac{\sin^2(\delta-\tau)}{\sin^2(\delta+\tau)}\right) \\
\text{oder} &= \frac{1}{2}\frac{\sin^2(\delta-\tau)}{\sin^2(\delta+\tau)}\left(1 + \frac{\cos^2(\delta+\tau)}{\cos^2(\delta-\tau)}\right) \\
\text{oder} &= \frac{1}{2}\left(\left(\frac{n_2\cos\delta + n_1\cos\tau}{n_2\cos\delta - n_1\cos\tau}\right)^2 + \left(\frac{n_1\cos\delta + n_2\cos\tau}{n_1\cos\delta - n_2\cos\tau}\right)^2\right) \\
\text{oder} &= \frac{1}{2}\frac{(g-c)^2}{(g+c)^2}\left(1 + \frac{(c(g+c)-1)^2}{(c(g-c)+1)^2}\right)
\end{aligned}$$

Wobei für die letzte Zeile gilt:

$$\begin{aligned}
c &= \cos\delta \\
g^2 &= \left(\frac{n_1}{n_2}\right)^2 + c^2 - 1
\end{aligned}$$

ρ ist natürlich von der Wellenlänge abhängig, da der Brechungsindex von dieser abhängt. Der Zusammenhang zwischen δ und τ ist über das SNELLsche Gesetz gegeben. In der zweiten oder–Gleichung wird der Winkel τ im Gegensatz zur bisherigen Definition auch gegen den Normalenvektor gemessen, der Winkel ist also größer als 90 Grad. Im andern Fall muß das Vorzeichen von $\cos\tau$ negiert werden.

Als Spezialfälle erhält man:

$$\delta = 0: \qquad \rho(0) = \left(\frac{n_2 - n_1}{n_2 + n_1}\right)^2$$

Ist das dünnere Medium Luft, vereinfacht sich die Gleichung weiter zu

$$\delta = 0, n_1 = 1 : \qquad \rho(0) = \left(\frac{n_2 - 1}{n_2 + 1}\right)^2$$

Hat man den Brechungsindex n in Abhängigkeit von der Wellenlänge gegeben, kann man ρ bestimmen. Sind Meßwerte für $\rho(0)$ vorhanden, kann man damit erst den Brechungsindex des Materials bestimmen und daraus schließlich $\rho(\delta)$ berechnen.

Das SNELLsche Gesetz und die FRESNELschen Gleichungen sind hier nur für den Fall von isotropen Medien angegeben. Handelt es sich um ein Material mit komplexem Brechungsindex, d.h. senkrecht und parallel zur Grenzfläche verlaufende Wellenanteile werden unterschiedlich gebrochen bzw. transmittiert, ergeben sich komplexe Ausdrücke. Diese lauten für den einfachen Fall, daß aus einem Medium mit reellem Brechungsindex $n \approx 1$ (Luft) eine Welle auf ein Medium mit komplexem Brechungsindex $n - \iota\kappa$ trifft (κ heißt auch Extinktionskoeffizient):

$$\begin{aligned}
\frac{\sin\tau}{\sin\delta} &= \frac{1}{n - \iota\kappa} \\
\rho_{\parallel} &= \frac{a^2 + b^2 - 2a\sin\delta\tan\delta + \sin^2\delta\tan^2\delta}{a^2 + b^2 + 2a\sin\delta\tan\delta + \sin^2\delta\tan^2\delta} \\
\rho_{\perp} &= \frac{a^2 + b^2 - 2a\cos\delta + \cos^2\delta}{a^2 + b^2 + 2a\cos\delta + \cos^2\delta}
\end{aligned}$$

wobei gilt:

$$\begin{aligned}
2a^2 &= \sqrt{(n^2 - \kappa^2 - \sin^2\delta)^2 + 4n^2\kappa^2} + (n^2 - \kappa^2 - \sin^2\delta) \\
2b^2 &= \sqrt{(n^2 - \kappa^2 - \sin^2\delta)^2 + 4n^2\kappa^2} - (n^2 - \kappa^2 - \sin^2\delta)
\end{aligned}$$

Der Reflexionskoeffizient ergibt sich bei unpolarisiert einfallendem Licht analog zu oben als Mittelwert der beiden Werte $\rho_{\parallel}$ und $\rho_{\perp}$.

Ist das Verhältnis $\kappa/n \ll 1$ kann weiter genähert werden:

$$a = n \qquad \text{und} \qquad b = k$$

womit sich obige Gleichungen weiter vereinfachen.

Im folgenden wird der Reflexionskoeffizient aufgrund der FRESNELschen Gleichungen mit F_{att} abgekürzt.

Übungen

1. Verifizieren Sie die oder–Zeilen der FRESNELschen Gleichungen.
2. Geben Sie an, welche Material–Parameter und welche geometrischen Informationen zur Berechnung der FRESNELschen Gleichungen benötigt werden. Unterscheiden Sie dabei gemäß den gemachten Näherungen.
3. Geben Sie ein möglichst billiges Berechnungsverfahren für die FRESNELschen Gleichungen an (z.B. als C–Code). Wählen Sie dabei Ihnen bekannte Rechenzeiten für die arithmetischen Operationen (sie können auch auf gemachte Messungen zurückgreifen). Unterscheiden Sie dabei gemäß den gemachten Näherungen.
4. Gegeben seien der Lichtvektor $\vec{L}$, die Brechungsindizes n_1 und n_2 sowie der Normalenvektor $\vec{N}$. Geben Sie möglichst einfache Verfahren zur Berechnung des Reflexionsvektors $\vec{R}$ und des Transmissionsvektors $\vec{T}$ an.

2.2 Reale Materialien

Die Theorie, die ansatzweise oben besprochen worden ist, beschreibt nur ideale Oberflächen. Reale Oberflächen sind jedoch auch durch folgende Eigenschaften gekennzeichnet:

- Sie bestehen aus keinem einheitlichen Material, dessen optischen Eigenschaften bekannt (?) sind.
- Sie sind nicht optisch glatt, d.h. die optischen Eigenschaften werden auch durch Mikro–Geometrie bestimmt.
- Es sind keine exakten Modelle zur Beschreibung der Materialien und der Oberflächen bekannt.
- Man kennt nicht genügend Meßdaten, um die obigen Formeln sinnvoll anwenden zu können.
- Die einfallende Strahlung wechselwirkt mit dem gesamten Material, d.h. sie dringt in das Material ein und unterliegt dort wiederum anderen physikalischen Gesetzen. (Diese Effekte sind insbesondere bei dünnen Farbfilmen auf Metallen zu beachten.)

Wir machen vorerst die Annahme, daß die Materialien nicht transparent sind.

2.2.1 Strahlungseigenschaften von Metallen

Metalle haben eine relativ geringe Emmisionsrate ϵ und damit auch eine geringe Absorptionsrate α. Demzufolge ist die Reflexionsrate ρ relativ hoch. Es gelten die folgenden Beziehungen:

- Mit steigendem Einfallswinkel δ steigt die Emmisionsrate ϵ und damit sinkt die Reflexionsrate ρ, d.h. je flacher man auf eine Metallplatte schaut, umso weniger spiegelt sie.
- Mit steigender Wellenlänge λ sinkt die Emmisionsrate ϵ und damit steigt die Reflexionsrate ρ. Kupfer macht hierbei eine Ausnahme: ϵ ist relativ konstant bezüglich λ.
- Mit steigender Temperatur T steigt die Emmisionsrate ϵ und damit sinkt die Reflexionsrate ρ.
- Für gewisse Wellenlängen– und Temperaturbereiche ist die Emmisionsrate proportional zum elektrischen Widerstand des Metalles.

2.2.2 Strahlungseigenschaften von Nicht–Metallen

Optische Materialeigenschaften von Nicht–Metallen sind noch weniger bekannt als die der Metalle. Nicht–Metalle verhalten sich oft umgekehrt wie Metalle, insbesondere sinkt die Emmisionsrate ϵ mit steigendem Einfallswinkel δ, d.h. schaut man flach auf eine Nicht–Metallplatte, spiegelt sie mehr als bei direkter Draufsicht. Die Wellenlängenabhängigkeiten sind deutlich geringer als die der Metalle. Dies gilt ebenso für die Temperaturabhängigkeit. M.a.W. der Brechungsindex ist annähernd konstant bezüglich λ und T.

Übungen

1. Suchen Sie in Tabellenbüchern nach optischen Materialeigenschaften (Hinweis: Es gibt Bibliotheken in der Physik, Elektrotechnik, NMI ...).

Beispiele aus LAX, E., D'ANS, *Taschenbuch für Chemiker und Physiker, Band 1, Springer Verlag, Berlin, 1967*: Brechungsindex von Gasen: $1 - 2 \ldots 700\ 10^{-6}$; Brechungsindex von anorganischen Verbindungen liegt meist in $1.3 \ldots 1.9$ selten bis 3.0 oder kleiner 1.3 (Bleisulfit sogar bis 3.9); Brechungsindex von Gläsern und Quarzen zwischen 1.4 und 1.7

2.3 Oberflächenrauheit

Eine Oberfläche kann — sofern sie keine Verwerfungen aufweist — als eine zweidimensionale Funktion $\zeta(x, y)$ aufgefaßt werden. Für einen Punkt (x, y) gibt $\zeta(x, y)$ die Abweichung der Oberfläche vom Mittelwert an. Es gilt somit: $\text{mean}(\zeta(x, y)) = 0$. Für die Standardabweichung σ_0 gilt:

$$\sigma_0 = \sqrt{\text{mean}(\zeta^2(x, y))} = \sqrt{\int\int \zeta^2(x, y)\ dxdy}$$

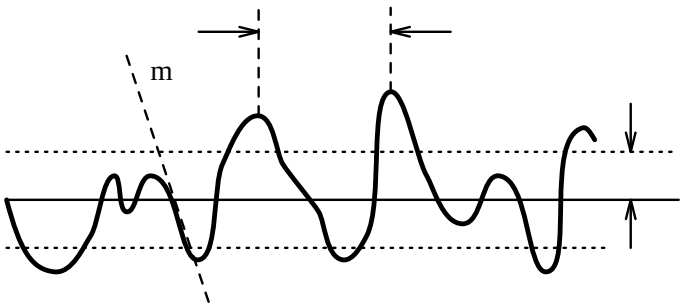

Abbildung: Rauhe Oberfläche

m	mittlere Steigung
τ	mittlere Berg/Talabstand (correlation distance)
σ_0	mittlere Abweichung vom Mittelwert

Für die Beziehungen zwischen den drei Parametern wird von BECKMANN (1963) aufgrund einer einfachen geometrischen Anschauung

$$m = \frac{2\sigma}{\tau}$$

und von BENNETT (1961)

$$\bar{m} = \frac{\sqrt{2}\sigma}{\tau}$$

angegeben. Die zweite Formel ergibt sich unter der Annahme, daß $\zeta(x, y)$ GAUSS–verteilt ist. Ein korrekter Wert ergibt sich nur aus der tatsächlichen Verteilung.

Nun kann man zwei Phenomene unterscheiden. Zum einen werden die Reflexionseigenschaften des Materials geändert, da die Oberfläche nicht mehr wie im theoretischen gefordert orientiert ist, was eine Reduktion der Reflexionsrate zur Folge hat. Dies wird durch zwei Reduktionsfaktoren D_{att} und G_{att} zum Ausdruck gebracht. Zum anderen wird möglicherweise reflektiertes oder einfallendes Licht durch die Oberfläche selbst abgeblockt — sogenannte Selbstbeschattung —, was ebenfalls die Reflexionsrate reduziert. Dies kann durch einen Abschwächungsfaktor S_{att} modelliert werden.

Bemerkung: In der Literatur wird S_{att} oft mit G_{att} bezeichnet, sowie $D_{att} * G_{att}$ als D_{att} zusammengefaßt.

2.3.1 Geometrische Selbstabschwächung

Wir wollen drei Abschwächungsfaktoren angeben.

- TORRANCE, SPARROW (1963) führten eine geometrische Selbstabschwächung nach einem einfachen Oberflächenmodell ein.

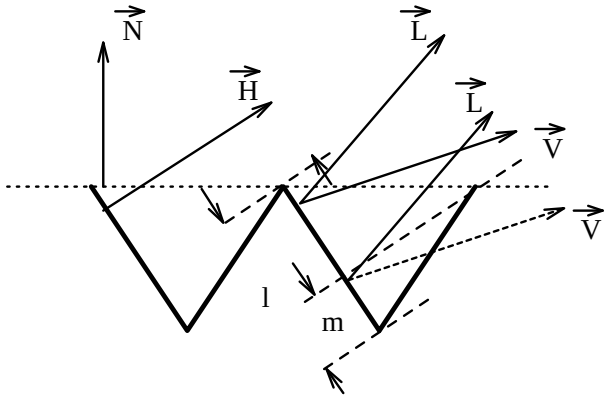

Abbildung: Selbstabschwächung

nach TORRANCE, SPARROW

Es ergibt sich:

$$S_{TS} = \min\left(1, \frac{2\langle\vec{N},\vec{H}\rangle\langle\vec{N},\vec{V}\rangle}{\langle\vec{V},\vec{H}\rangle}, \frac{2\langle\vec{N},\vec{H}\rangle\langle\vec{N},\vec{L}\rangle}{\langle\vec{V},\vec{H}\rangle}\right)$$

Dieser Faktor hat die Nachteile, daß er nicht stetig und nicht symmetrisch ist,

- was bei dem Faktor nach SANCER (1969), der mithilfe von Normalverteilungen hergeleitet ist, nicht der Fall ist.

$$S_S = \frac{1}{1 + C_0 + C_1}$$

Dabei sind die Werte für C_0 und C_1 relativ kompliziert zu berechnen:

$$\begin{aligned}
C_i &= \frac{1}{2}\left(\frac{e^{-c_i}}{\sqrt{\pi c_i}} - 1 + \Phi(\sqrt{c_i})\right) \\
c_i &= \frac{\tan\delta_i}{2\bar{m}} \\
\delta_0 &= \arccos(\langle\vec{N},\vec{L}\rangle) \\
\delta_1 &= \arccos(\langle\vec{N},\vec{V}\rangle) \\
\Phi(x) &= \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{x} e^{\frac{x^2}{2}}\,dx
\end{aligned}$$

d.h. $\Phi(x)$ berechnet das GAUSSsche Fehlerintegral. Dies ist in mathematischen Bibliotheken (C–Compiler) als `erf()` enthalten. Auch die Funktion $1 - \Phi(x)$ ist als `erfc()` vorhanden. Für den SANCER–Faktor ist eine weitere Materialeigenschaft $\bar{m}$ notwendig, was bei TORRANCE, SPARROW nicht erforderlich ist.

- TORRANCE, GREENBERG verwenden einen Faktor nach SMITH, BRUCE, der noch komplizierter zu berechnen ist:

$$\begin{aligned}
S_{SB} &= S_0 S_1 \\
S_i &= \frac{1 - \frac{1}{2}(1 - \Phi(c_i))}{\frac{1}{2}(\frac{1}{c_i\sqrt{\pi}} - (1 - \Phi(c_i))) + 1} \\
c_i &= \frac{\tau}{2\sigma_0 \tan\delta_i}
\end{aligned}$$

(δ_i und Φ wie oben.) Hier kann auch wiederum m eingesetzt werden.

2.3.2 Verteilungsabschwächung

Die Oberfläche wird nicht mehr als einheitliche Fläche, sondern als Ansammlung von Micro–Facetten beschrieben, von denen nur diejenigen zu einer Reflexion beitragen, die entsprechend orientiert sind. Es kommt also auf die Verteilung und die geometrische Anordnung der Micro–Facetten innerhalb der Oberfläche an. Dies kann durch zwei Faktoren D_{att} und G_att modelliert werden. Als mögliche Verteilungen wurden die folgenden vorgeschlagen:

- PHONG

$$\begin{aligned} D_P &= \langle \vec{N}, \vec{H} \rangle^{Ns} \\ G_P &= 1 \end{aligned}$$

  Diese Verteilung wird in den meisten einfachen Modellen (siehe Beschreibung der Modelle) verwendet. Sie ist für das Entstehen von highlights verantwortlich.

- TORRANCE, SPARROW (1967) Auch diese Verteilung ist recht einfach, wurde jedoch selten (?) benutzt.

$$\begin{aligned} D_{TS} &= e^{-(C_1 \arccos\langle \vec{N}, \vec{H} \rangle)^2} \\ G_{TS} &= 1 \end{aligned}$$

- TROWBRIDGE, REITZ (1967) Eine weitere Verteilung ist

$$\begin{aligned} D_{TR} &= \left(\frac{C_2^2}{\langle \vec{N}, \vec{H} \rangle^2 (C_2^2 - 1) + 1} \right)^2 \\ G_{TR} &= 1 \end{aligned}$$

  Diese ersten drei Verteilungen sind sich sehr ähnlich und können annähernd ineinander umgerechnet werden, was später bei der Beschreibung der Modelle erläutert wird.

- BECKMANN, BAHAR (1963,1987) Die folgenden Verteilungen nutzen eine statistische Beschreibung der Oberflächeneigenschaften.

$$GD_{BB} = \begin{cases} \frac{1}{4\pi m^2 \cos^4 \alpha} g^2 e^{-\left(g + \frac{v_{xy}^2 \tau^2}{4}\right)} & \text{falls } g \ll 1 \\ \frac{1}{4\pi m^2 \cos^4 \alpha} \sum_{i=1}^{\infty} \frac{g^{i+1}}{i!\, i} e^{-\left(g + \frac{v_{xy}^2 \tau^2}{4i}\right)} & \text{falls sonst} \\ \frac{1}{4\pi m^2 \cos^4 \alpha} e^{-\frac{\tan^2 \alpha}{m^2}} & \text{falls } g \gg 1 \end{cases}$$

  wobei

$$\begin{aligned} g &= \frac{4\pi^2 \sigma^2}{\lambda^2} (\cos \delta_0 + \cos \delta_1)^2 \\ v_{xy}^2 &= \frac{4\pi^2}{\lambda^2} (\sin^2 \delta_0 - 2 \sin \delta_0 \sin \delta_1 \cos \varphi + \sin^2 \delta_1) \\ \cos \alpha &= \langle \vec{N}, \vec{H} \rangle \end{aligned}$$

  Der erste Term ergibt sich aus dem zweiten, wenn man nur das Anfangsglied der Summe berücksichtigt. Der letzte Term ergibt sich durch eine Grenzwertbetrachtung der Summe. Unter Berücksichtigung von $m = 2\sigma/\tau$ kann der zweite Term wie folgt umgeschrieben werden:

$$GD_{BB} = \frac{1}{4\pi m^2 \cos^4 \alpha} \sum_{i=1}^{\infty} \frac{g^{i+1}}{i!\, i} e^{-\left(g + \frac{v_{xy}^2 \tau^2}{4i}\right)}$$

$$
\begin{aligned}
&= \frac{g\tau^2}{4\pi 4\sigma^2 \cos^4\alpha} \sum_{i=1}^{\infty} \frac{g^i}{i!\ i} e^{-\left(g+\frac{v_{xy}^2 \tau^2}{4i}\right)} \\
&= \frac{4\pi^2\sigma^2(\cos\delta_0+\cos\delta_1)^2\tau^2}{\lambda^2 4\pi 4\sigma^2\cos^4\alpha} \sum_{i=1}^{\infty} \frac{g^i}{i!\ i} e^{-\left(g+\frac{v_{xy}^2 \tau^2}{4i}\right)} \\
&= \frac{(\cos\delta_0+\cos\delta_1)^2}{4\cos^4\alpha} * \frac{\pi\tau^2}{4\lambda^2} \sum_{i=1}^{\infty} \frac{g^i}{i!\ i} e^{-\left(g+\frac{v_{xy}^2 \tau^2}{4i}\right)} \\
&= G_{BB} D_{BB}
\end{aligned}
$$

- GREENBERG, TORRANCE erweitern diese Betrachtungsweise indem sie einerseits nicht nur σ in dieser einfachen Form betrachten, sondern dieses ebenfalls von den Winkeln abhängig betrachten, also eine effektive Rauheit beschreiben, und andererseits den geometrischen Faktor weitergehend analysieren, so daß sogar Polarisationseigenschaften modelliert werden können. Wir werden hier allerdings nur die vereinfachte Form angeben, die für nicht polarisiertes, einfallendes Licht gilt.

  Der Verteilungsabschwächungsfaktor ist analog dem der zweiten Gleichung von BECKMANN, BAHAR D_{BB}:

  $$D_{GT} = \frac{\pi\tau^2}{4\lambda^2} \sum_{i=1}^{\infty} \frac{g^i}{i!\ i} e^{-\left(g+\frac{v_{xy}^2 \tau^2}{4i}\right)}$$

  allerdings wird bei der Berechnung von g

  $$g = \frac{4\pi^2\sigma^2}{\lambda^2}(\cos\delta_0+\cos\delta_1)^2$$

  nicht die einfache Beziehung für σ eingesetzt, sondern auch hier wird eine effektive Rauheit eingesetzt:

  $$\sigma = \frac{\sigma_0}{\sqrt{1+\frac{z_0^2}{\sigma_0^2}}}$$

  wobei sich z_0 aus der Lösung der Gleichung

  $$\sqrt{\frac{\pi}{2}} z_0 = \frac{\sigma_0}{4}(K_0+K_1) e^{-\frac{z_0^2}{2\sigma_0^2}}$$

  ergibt. Die beiden Werte K_0 und K_1 berechnen sich für den einfallenden und den reflektierenden Vektor als:

  $$K_i = \tan\delta_i (1-\Phi(\frac{\tau}{2\sigma_0\tan\delta_i}))$$

  wobei die δ_i — analog zu oben — die entsprechenden Winkel darstellen.

  Der geometrische Abschwächungsfaktor wird wie folgt angegeben:

  $$G_{TG} = \left(\frac{\langle\vec{v},\vec{v}\rangle}{v_z}\right)^2 \frac{(\langle\vec{s_v},\vec{L}\rangle^2+\langle\vec{p_v},\vec{L}\rangle^2)(\langle\vec{s_l},\vec{V}\rangle^2+\langle\vec{p_l},\vec{V}\rangle^2)}{|\vec{V}\times\vec{L}|^4}$$

  hierbei sind die Vektoren $\vec{v}$, $\vec{s_v}$ und $\vec{p_v}$

  $$
  \begin{aligned}
  \vec{v} &= \vec{L}+\vec{V} \\
  \vec{s_v} &= \vec{V}\times\vec{N} \\
  \vec{p_v} &= \vec{s_v}\times\vec{V}
  \end{aligned}
  $$

  Die beiden anderen Vektoren $\vec{s_l}$ und $\vec{p_l}$ werden analog mithilfe von $\vec{L}$ berechnet.

Übungen

1. Leiten die S_{TS} mithilfe von geometrischen Betrachtungen her.
2. Seine $\vec{V}$ und $\vec{L}$ normiert. Zeigen Sie, daß gilt:

$$\begin{aligned} \vec{v} &:= \vec{V} + \vec{L} \\ v_{xy}^2 &= \frac{4\pi^2}{\lambda^2}(v_x^2 + v_y^2) \end{aligned}$$

3. Welchen Einfluß auf die Darstellung haben die Abschwächungsfaktoren. Erläutern Sie das Aussehen der Oberflächen in Abhängigkeit der Materialparameter.

2.4 Modelle

Zur besseren Gliederung und einfacheren Referenzierung werden im folgenden die Modelle in der Reihenfolge ihres Auftretens numeriert. Die Modelle sollten nicht nur für einen speziellen Darstellungsalgorithmus angesehen werden, sondern sie können je nach erforderlichen Parameter für die verschiedenen Verfahren genutzt werden.

Übungen

1. Kennzeichnen Sie die Vor– und Nachteile der folgenden Modelle im Hinblick auf die Darstellungsmöglichkeiten realer Szenen.

Generell sei bemerkt, daß eine Lichtquelle in Richtung $\vec{L}$ nur dann einen Beitrag zur Intensität an einer Oberfläche mit Normalenvektor $\vec{N}$ leistet, wenn gilt:

$$\cos\delta = \langle \vec{N}, \vec{L} \rangle \geq 0$$

d.h. im folgenden wird, wenn es nicht explizit anders gesagt wird, das Skalarprodukt zweier Vektoren gleich Null gesetzt, wenn es negativ sein sollte.

2.4.1 Modelle für ambiente Reflexion

Modell 1 (BOUKNIGHT 1970)

$$I_a = k_a\, I$$

Das ambiente Modell bleibt mithilfe des Superpositionsprinzip in den Darstellungsmodellen erhalten, d.h. dort wird ein Teil der Intensität aufgrund obiger Vorschrift berechnet. Dies garantiert eine Grundhelligkeit in der darzustellenden Szene, da gerade viele Verfahren die indirekte, diffuse Beleuchtung von Oberflächen durch Reflexionen von anderen Objekten ungenügend modellieren. Eine Ausnahme bildet das Radiosity–Verfahren, das speziell diesen Aspekt der Beleuchtung berücksichtigt.

Vorgreifend auf die abschließende Zusammenfassung sei hier schon gesagt, daß alle Beleuchtungsmodelle (wieder mit Ausnahme von Radiosity) die Intensität an einem Punkt gemäß folgender Vorgehensweise berechnen:

$$I = I_a + I_d + I_s + I_t$$

wobei I_a den ambienten Intensitätsanteil, I_d den aufgrund diffuser Reflexionseigenschaften gegebenen Intensitätsanteil, I_s den aufgrund von spiegelnder Reflexionseigenschaften gegebenen Intensitätsanteil und I_t den aufgrund der Transparenz der Oberfläche entstehenden Intensitätsanteil beschreibt. Es wird also in allen Fällen das einfache Superpositionsprinzip angewendet.

COOK, TORRENCE führten einen durch die Geometrie der Szene bestimmten Abschwächungsfaktor f ein, der den ambienten Intensitätsanteil beeinflussen soll. (k_a ist ein Objektparameter und I_a ein der gesamten Szenen gegebener Parameter.)

Modell 2 (COOK, TORRANCE 1981)

$$I_a = k_a \; f \; I$$

Allerdings wird dessen Berechnung nicht weiter ausgeführt. Man könnte ihn als eine frühe Vorwegnahme des Radiosity–Ansatzes interpretieren.

2.4.2 Modelle für diffuse Reflexion

Dieses Modell berücksichtigt keine Position des Betrachters, d.h. die darzustellende Intensität ist nicht vom Betrachterwinkel $\vec{V}$ abhängig.

Modell 3 (BOUKNIGHT 1970)

$$I_d = k_d \; \langle \vec{N}, \vec{L} \rangle \; I_L$$

was bei punktförmigen Lichtquellen zu

$$I_d = k_d \; \langle \vec{N}, \vec{L} \rangle \; I_L$$

führt und bei unendlich weit entfernten Punktquellen, d.h. parallelem Lichteinfall, wie folgt vereinfacht werden kann

$$I_d = k_d \; \langle \vec{N}, \vec{E_z} \rangle \; I_L$$

wenn durch Koordinatentransformation der Lichtvektor $\vec{L} = (0, 0, 1)^T$ ist.

Sind l Lichtquellen vorhanden, erhält man durch Superposition:

$$I_d = k_d \; \sum_{i=1}^{l} \langle \vec{N}, \vec{L_i} \rangle \; I_{Li}$$

Dieses Superpositionsprinzip kann sinngemäß auf alle folgenden Modelle angewandt werden und wird vereinfachend nicht stets explizit angegeben. In der zusammenfassenden Tabelle am Ende des Kapitels erscheint dann jeweils die Berechnung mit mehreren Lichtquellen.

Bis jetzt wurde die Betrachterposition sowie die Entfernung zur Lichtquelle noch nicht berücksichtigt.

2.4.3 Modelle für diffuse Reflexion mit Entfernungsberücksichtigung

An dieser Stelle seien zwei frühe, einfache, empirisch begründete Modelle angeführt, die zur Darstellung von Grau–Szenen verwendet worden sind, mehr wohl mit dem Hintergrund, Ergebnisse von Darstellungsalgorithmen zu dokumentieren. Insbesondere ist die Intensität einer Oberfläche nicht von der Entfernung der Oberfläche von der Lichtquelle, sondern von der Entfernung vom Betrachter abhängig.

Modell 4 (WARNOCK 1969)

$$I_d = \frac{|\langle \vec{N}, \vec{L} \rangle|}{|\vec{V}|}$$

Modell 5 (ROMNEY 1969)

$$I_d = k \, \frac{\langle \vec{N}, \vec{L} \rangle^2}{|\vec{V}|^4}$$

Hier wurden die Flächen auch von hinten gezeichnet; es gab anscheinend noch keine Objekte im heutigen Sinn. Hier kann das Skalarprodukt auch negativ sein.

Die Lichtquellen wurden bisher als Punktlichtquellen modelliert, was keine Berechnung von Raumwinkeln, aus denen ein Objekt beleuchtet wird, erlaubt. Insbesondere sinkt die Intensität mit wachsender Entfernung von der Lichtquelle mit $1/r_0^2$, wie wir oben gesehen haben.

Modell 6 (COOK, TORRANCE 1981)

$$I_d = k_d \, \langle \vec{N}, \vec{L} \rangle \, d\Omega \, I_L$$

Dabei ist $d\Omega$ der Raumwinkel, unter dem die Lichtquelle von der Fläche aus gesehen wird.

Meine Interpretation: Modelliert man die Lichtquellen als Kugeln und sind sie weit von der bestrahlten Fläche entfernt (Abstand $|\vec{L}|$), kann man die Näherung

$$d\Omega = \frac{dS}{|\vec{L}|^2} \cos \delta_S$$

anwenden. Es gilt weiterhin

$$\cos \delta_S = 1$$

da die Lichtquelle eine Kugel ist. Wählt man den Radius der Kugeln so, daß für die Fläche $dS = 1$ gilt oder integriert man die Größe der Lichtquelle in deren Intensität I_L, erhält man

Modell 7

$$I_d = \frac{k_d \, \langle \vec{N}, \vec{L} \rangle \, I_L}{|\vec{L}|^2}$$

Nun kann man auch das Abschwächen der Strahlung auf dem Weg der Länge $|\vec{V}|$ vom Objekt zum Betrachter berücksichtigen:

Modell 8 (VAM DAM 1984)

$$I_d = \frac{k_d \ \langle \vec{N}, \vec{L} \rangle \ I_L}{(|\vec{L}| + |\vec{V}|)^2}$$

Meine Interpretation: Eigentlich müßte die Entfernung zum Betrachter wie folgt in das Modell eingebaut werden:

Modell 9

$$I_d = \frac{k_d \ \langle \vec{N}, \vec{L} \rangle \ I_L}{|\vec{L}|^2 |\vec{V}|^2}$$

Da die reflektierte Intensität — sozusagen, ab dem Punkt ihrer Erzeugung — quadratisch mit dem Abstand vom Betrachter abnimmt. Dieses belegt auch möglicherweise die frühen, guten Ergebnisse von ROMNEY (Modell 5).

Modell 6 hat den Nachteil, daß der Raumwinkel $d\Omega$ der Lichtquelle bekannt sein muß. Verwendet man die Näherung (Modell 7) oder auch die Modelle 8 und 9, die jeweils den Abstand berücksichtigen, ergeben sich zwei Probleme: bei unendlich weit entfernten Lichtquellen kann die Gleichung nicht verwendet werden, bei geringem Abstand zwischen Lichtquelle und Objekt ist die Näherung nicht gültig und man erzielt schlechte Resultate. Deshalb wurden weiter folgende Lösungen vorgeschlagen:

Modell 10 (FOLEY, VAN DAM 1984 ?)

$$I_d = \frac{k_d \ \langle \vec{N}, \vec{L} \rangle \ I_L}{|\vec{V}| + d_k}$$

Die Modelle 7 bis 10 kann man wie folgt zusammenfassen:

Modell 11 (FOLEY, VAN DAM 1990)

$$I_d = k_d \ d_{att} \ \langle \vec{N}, \vec{L} \rangle \ I_L$$

wobei d_{att} ein sogenannter Abschwächungsfaktor (attenuation factor) ist; hier bezüglich der Entfernung. Mit Hilfe von Abschwächungsfaktoren können auch andere Effekte, wie Atmospheren u.a., modelliert werden. Weitere Möglichkeiten der Beeinflussung der Eigenschaften der Lichtquelle werden in späteren Abschnitten erläutert.

(Hier würde es mit Radiosity weitergehen.)

2.4.4 Modelle für spiegelnde Reflexion

Eines der ersten, frühen Modelle ist:

Modell 12 (WARNOCK 1969 ?)

$$I_s = \frac{\langle \vec{N}, \vec{L} \rangle^m}{|\vec{V}|}$$

Lichtquelle und Betrachter müssen im gleichen Punkt liegen. Eine Verbesserung wurde von PHONG vorgeschlagen:

Modell 13 (PHONG 197?)

$$I_s = k_s \, \langle \vec{R}, \vec{V} \rangle^m \, I_L$$

Die Größe der Materialkonstanten m bestimmt die Größe eines highlights auf der bestrahlten Fläche. Ursprünglich war k_s ein vom Einfallswinkel δ abhängiger Materialparameter, der empirisch bestimmt wurde. FOLEY, VAN DAM benutzt allerdings später nur eine Konstante und erhält trotzdem ßufriedenstellende"Bilder, allerdings wird wiederum die Entfernung vom Betrachter eingebracht:

Modell 14 (FOLEY, VAN DAM 1984 ??)

$$I_s = \frac{k_s \, \langle \vec{R}, \vec{V} \rangle^m \, I_L}{|\vec{V}| + d_k}$$

Um nun nicht den Reflexionsvektor $\vec{R}$ berechnen zu müssen, kann man ihn durch den Halbvektor $\vec{H}$ und den Betrachtervektor $\vec{V}$ durch den Normalenvektor $\vec{N}$ ersetzen. Es gilt also:

$$\vec{H} = \frac{1}{2}(\vec{L} + \vec{V})$$

Modell 15 (BLINN 1977)

$$I_s = k_s \, \langle \vec{H}, \vec{N} \rangle^m \, I_L$$

Analog kann man wiederum den Abstand integrieren:

Modell 16 (FOLEY, VAN DAM 1984 ??)

$$I_s = k_s \frac{\langle \vec{H}, \vec{N} \rangle^m \, I_L}{|\vec{V}| + d_k}$$

YANG führte ein modifiziertes Modell nach PHONG ein, bei dem der highlight–Faktor geringfügig anders berechnet wird:

Modell 17 (YANG 1987)

$$I_s = k_s \left(\frac{\langle \vec{V}, \vec{R} \rangle + 1}{2} \right)^m I_L$$

Übungen

1. Erklären Sie, weshalb man den Vektor $\vec{H}$ verwenden kann ? (Hinweis: Interpretieren Sie ihn als Normalenvektor einer Ebene.)
2. Bestimmen Sie den genauen Fehler, den man durch die Näherung mit dem Vektor $\vec{H}$ in Modell 15 gegenüber Modell 14 macht. Kann man ihn durch geeignete Wahl der Materialparameter m und k_s annähernd kompensieren ?
3. Motivieren Sie die Modifikation von YANG.

2.4.5 Modelle mit Berücksichtigung der Oberflächenrauheit

Die oben gemachte Modellierung der Oberflächeneigenschaften kann wie folgt in die Berechnung der Intensität integriert werden:

Modell 18 (BLINN 1977)

$$I_s = k_s \frac{D_{att} G_{att} F_{att}}{\langle \vec{N}, \vec{V} \rangle} I_L$$

Dabei benutzt BLINN $G_{att} = G_{TS}$ und für D_{att} die Verteilungen D_P, D_{TS} und D_{TR}. Hierbei stellt er fest, daß die Verteilungen sich sehr ähnlich sind. Fordert man, daß $D_{att} = \frac{1}{2}$ für alle drei Verteilungen an der gleichen Stelle $\alpha = \arccos\langle \vec{N}, \vec{H} \rangle$ ist, kann man die Materialkonstanten Ns, C_1 und C_2 auf einen einzigen Winkel β als Materialkonstante zurückführen:

$$\begin{aligned} Ns &= -\frac{\ln(2)}{\ln(\cos\beta)} \\ C_1 &= \frac{\sqrt{\ln(2)}}{\beta} \\ C_2 &= \sqrt{\frac{\cos^2\beta - 1}{\cos^2\beta - \sqrt{2}}} \end{aligned}$$

Modell 19 (COOK, TORRANCE 1982)

$$I_s = k_s \frac{D_{BB} G_{TS} F_{att}}{\langle \vec{N}, \vec{V} \rangle} I_L$$

Auch hier kann die Konstante m in D_B wieder auf den Winkel β zurückgeführt werden:

$$m^2 = -\frac{\tan^2\beta}{\ln(\frac{1}{2}\cos^4\beta)}$$

Test meinerseits lassen jedoch die Formel

$$m = \frac{1}{2\sqrt{\pi}}(1 + \sqrt{\tan \beta})$$

besser erscheinen.

Dieses Modell geht davon aus, daß die Oberflächen relativ rauh gegenüber der Wellenlänge des Lichtes sind, da nur die letzte Gleichung der BECKMANNschen Gleichungen verwendet worden ist. Insbesondere heißt dies, daß man die Verteilung nicht für sehr glatte, spiegelnde Oberflächen einsetzen sollte.

Für die geometrische Abschwächung kann auch die Funktion G_S nach SANCER in den Modellen 18 und 19 eingesetzt werden.

2.4.6 Neuestes Modell

Torrance, Greenberg führten 1991 ein neues, physikalisch begründetes Modell ein, das auch in der Lage ist, Polarisationseffekte zu beschreiben. Auf die Polarisationseffekte wird hier nicht eingegangen, sondern lediglich das vereinfachte Modell wiedergegeben, das die Abschwächungsfaktoren, wie sie weiter oben erläutert worden sind, verwendet. Ein wesentlicher Punkt in diesem Modell ist die Tatsache, daß der spiegelnde Intensitätsanteil erneut in zwei Anteile aufgespalten worden ist. Für den diffusen Lichtanteil I_d wird weiterhin das Modell 6 benutzt.

Modell 20 (TORRANCE, GREENBERG 1991)

$$\begin{aligned} I_s &= I_{ss} + I_{sd} \\ I_{ss} &= k_{ss} F_{att} e^{-g} S_{SB} \Delta I_L \\ I_{sd} &= k_{sd} \frac{D_{TG} G_{TG} F_{att}}{\langle \vec{N}, \vec{V} \rangle} I_L \end{aligned}$$

S_{SB} ist die Selbstbeschattungsfunktion nach SMITH, BRUCE. Δ ist eine Delta–Funktion, die entscheided, ob der betrachtete Strahl im sogenannten Spiegelkegel liegt, d.h. einem kleinen Raumwinkel, indem man spiegelnde Reflexion erwartet. Man kann Δ z.B. wie folgt berechnen:

$$\Delta = \begin{cases} 1 & \langle \vec{V}, \vec{R} \rangle \leq \gamma \\ 0 & \text{sonst} \end{cases}$$

wobei der Winkel γ entweder eine Materialkonstante oder auch eine Szenenkonstante sein kann.

2.4.7 Modelle für Transparenz

Das einfachste Modell berücksichtigt keine Brechungseffekte und macht die Annahme von unendlich dünnen Oberflächen:

Modell 21

$$I = (1 - k_t)I_t + k_t\ I_r$$

Dabei ist I_t die Intensität, die durch das im Hintergrund liegende Objekt erzeugt wird. I_r ist der Teil der Intensität, der ohne Transparenz berechnet würde. Man erkennt, das dieses Modell einfach eine lineare Interpolation der beiden Intensitäten ist. k_t gibt die Transmissionsrate an.

In einfachen Darstellungsalgorithmen kann man die Transparenzrate von der Dicke des Objektes in Blickrichtung (vorallem bei parallel Projektionen) abhängig machen. Dies wurde von CROW wie folgt vorgeschlagen:

Modell 22 (CROW 1978)

$$k_t = (k_{tmax} - k_{tmin})(1 - (1 - N_z)^p) + k_{tmin}$$

Die Berechnung der Intensität bleibt analog zu Modell 20. Hierbei ist vereinfachend davon ausgegangen, daß der Betrachter in z–Richtung blickt.

Die Modellierung von Transparenz ist wesentlich schwieriger als die von Reflexion und wurde auch nicht so stark verfolgt wie diese. Das folgende Modell beinhaltet viele Vereinfachungen, kann jedoch für einzelne Strahlverfolgungen nach dem SNELLschen Gesetz eingesetzt werden.

Modell 23 (HALL 1981)

$$k_t = \frac{1 - \left(\frac{n-1}{n+1}\right)^2}{1 + \left(\frac{n-1}{n+1}\right)^2}$$

Der Brechungsindex ist von der Wellenlänge abhängig, was man für Dispersion ausnutzen kann. Bei transparenten Materialien muß man berücksichtigen, daß beim Verlauf der Strahlung durch das Material eine Abschwächung a_{att} — analog zu einem atmospherischen Effekt — auftritt, die ebenfalls von der Wellenlänge abhängig sein kann. So ist zum Beispiel in Wasser die Absorptionsrate von grünem Licht am geringsten (siehe Übung).

1. Schlagen Sie einen Abschwächungsfaktor a_{att} vor, der die Intensitätsabnahme in einem Medium modelliert. Beschränken Sie sich dabei auf eine konstante Absorptionsrate des Mediums, d.h. berücksichtigen Sie keine Streuungen.

2.4.8 Zusammenfassung

Durch Superposition und je nach verwendetem Darstellungsalgorithmus können also die folgenden Lichtanteile, die je nach gewünschtem Modell berechnet werden, überlagert werden:

I_a	ambienter Lichtanteil
I_d	diffuser Lichtanteil
I_s	durch Reflexion erzeugter Lichtanteil (in Modell 20 wiederum aufgespalten)
I_t	durch Transparenz erzeugter Lichtanteil

In den Modellen wurden folgende Abschwächungsfaktoren eingeführt, die je nach Modell unterschiedliche Materialparameter notwendig machen:

F_{att}	FRESNEL–Abschwächung
D_{att}	Verteilungsabschwächung
G_{att}	geometrische Abschwächung
S_{att}	Abschwächung durch Selbstbeschattung
d_{att}	Entfernungsabschwächung
a_{att}	Absorptionsabschwächung

2.5 Geometrie von Lichtquellen

Unter den geometrischen Eigenschaften von Lichtquellen ist weniger der tatsächliche Aufbau der Lichtquelle gemeint, sondern vielmehr die Verteilung der Intensität in den umgebenden Raum. Dies löst oft in empirischer Hinsicht das Problem, die Intensitätsabschwächung bzgl. des Abstandes der strahlungserzeugenden von der strahlungsempfangenden Fläche bzw. einfache Eigenschaften einer Atmosphere in die Modellierung gut einzubeziehen.

empirische Licht–Abschwächung FOLEY, VAN DAM (1991) schlagen desweiteren den folgenden Abschwächungsfaktor vor:

$$l_{att} = \min\left(\frac{1}{c_1 + c_2|\vec{L}| + c_3|\vec{L}|^2}, 1\right)$$

wobei die Konstanten Parameter der Lichtquelle sind.

WARN–Lichtquellen WARN (1983) modelliert eine Lichtquelle mithilfe des PHONGschen Cosinus–Terms. Hierbei wird zusätzlich zum Ort der Lichtquelle ein Normalenvektor $\vec{N}_L$ definiert. Die Intensität ergibt sich dann:

$$I_L = I_{max}\ \langle \vec{L}, \vec{N}_L \rangle^p$$

dabei ist p wiederum ein Parameter, der den Durchmesser des spot lights festlegt.

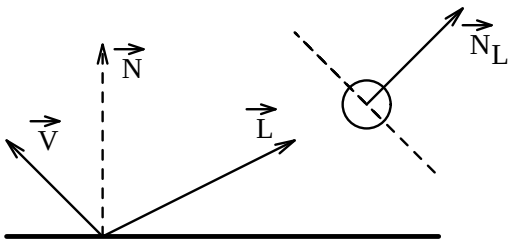

Abbildung: PHONG–Lichtquelle

Es besteht auch die Möglichkeit die Intensität des Lichtes mithilfe eines binären Faktors b_L auf bestimmte Regionen des Raumes einzuschränken. Als Beispiele werden von WARN folgende Regionen genannt: Kegel, Spalten (Räume zwischen zwei Ebenen), Würfel um die Punktlichtquelle.

Fading oder depth cueing Um beispielsweise die Intensität einer Lichtquelle einfach mit der Entfernung zu variieren kann man folgende Gleichung verwenden:

$$I_L = \begin{cases} I_s & \text{falls } d \leq s \\ \frac{e-d}{e-s} I_s + \frac{d-s}{e-s} I_e & \text{falls } s < d < e \\ I_e & \text{falls } d \geq e \end{cases}$$

Dabei sind s und e Start– bzw. Endwert einer linearen Fade–Funktion. d ist der Abstand zur Lichtquelle $|\vec{L}|$ oder der Abstand zum Betrachter $|\vec{V}|$ je nach gewählter Modellierung. Dies kann in einen Abschwächungsfaktor integriert werden.

Übungen

1. Geben Sie für folgende Regionen um die Lichtquelle die entsprechenden Berechnungsvorschriften für b_L an: Kegel, Spalte parallel zum Weltkoordinatensystem, offener Zylinder in beliebiger Richtung, Kugel um die Punktlichtquelle.

2. Kennen Sie weitere sinnvolle Regionen, die man für die Modellierung nach WARN von Lichtquellen benutzen kann ?

3. Suchen Sie nach Möglichkeiten, die Grenzen der Regionen nicht als scharfe Intensitätssprünge erscheinen zu lassen.

3 RayTracing

Literatur:

- RAUBER, THOMAS; Algorithmen in der Computer Graphik; *erscheint in Springer Verlag, voraus. 1993*
- FOLEY, JAMES; VAN DAM, ANDRIES; Computer Graphics, Principles and Practice; *2nd Edition, Addison Wesley, 1991*
- GLASSNER ANDREW; An Introduction to Ray Tracing; *Academic Press, San Diego, USA, 1989*
- SPEER, L.RICHARD; An Updated Cross–Indexed Guide to the Ray–tracing Literature; *erhältlich über anonymous ftp*
- GILL, CHRISTIAN; Implementierung von parallelem Ray–Tracing auf DATIS–P–32; *Diplomarbeit, Universität des Saarlandes, 1992*

RayTracing steht für Strahlverfolgung, d.h. einem Darstellungsalgorithmus mit Beleuchtungsverfahren zur realistischen Wiedergabe von Szenen. Es ist besonders gut geeignet für die Darstellung von:

- Schattenwurf
- Transparenz mit Brechung
- spiegelnder Reflexion

Die Schwächen liegen in den folgenden Punkten:

- diffuse Reflexion und damit verbunden weiche Schattenverläufe
- indirekte Beleuchtung, sowohl diffuse als auch spiegelnd
- Berechnungsaufwand

3.1 Die Idee des RayTracings

Da es nicht praktikabel ist, alle ausgesendeten Strahlen zu verfolgen, geht man den umgekehrten Weg. Die potentiell ins Auge einfallenden Lichtstrahlen (Teilchenmodell, Strahlenoptik (siehe Kapitel 2)) werden zurückverfolgt. Ein Schnitt im entstehenden Strahlkegel wird als Projektionsfläche dargestellt. Treffen die Strahlen auf Oberflächen, werden sie entsprechend den optischen Gesetzen (Reflexion, Transparenz, Brechung usw.) gespiegelt bzw. gebrochen. Die Strahlen transportieren die Energie (Farbe und Intensität, je nach Farbmodell RGB, YIQ (YUV), HSL usw.) des Lichtes. Die von einer Lichtquelle ausgesendete Energie wird so für den in das Auge einfallenden Strahl berechnet. Die Energie wird ebenfalls durch die Materialeigenschaften manipuliert (siehe Kapitel 2).

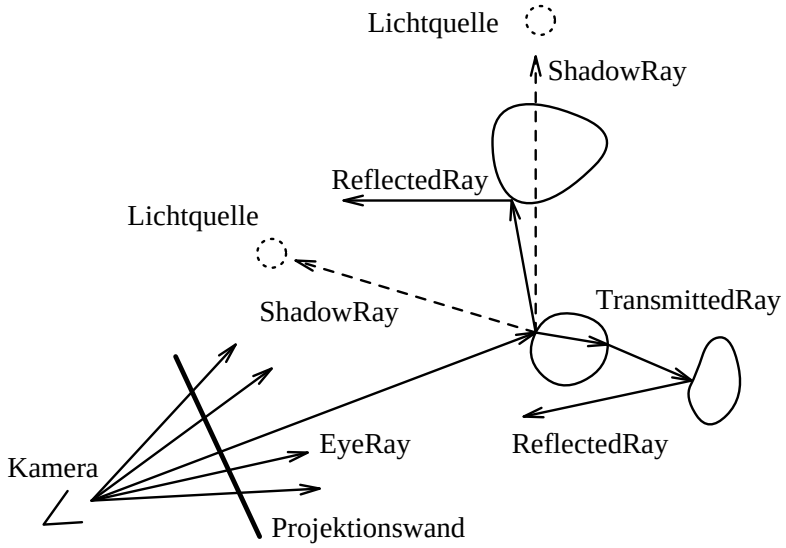

Abbildung: RayTrace–Algorithmus

Obige Abbildung veranschaulicht die Vorgehensweise. Im folgenden wird auf die dort gemachten Bezeichnungen für die verschiedenen Strahlen zurückgegriffen. Der Algorithmus arbeitet also wie folgt:

- von der Kamera wird ein Primärstrahl oder EyeRay ausgesandt;
- trifft er ein Objekt werden
  - Schattenfühler (LightSensor oder ShadowRay) zu allen Lichtquellen gesandt; liegt der betrachtete Punkt nicht im Schatten, wird die Intensität gemäß des Beleuchtungsmodells berechnet;
  - der ReflectedRay wird ausgesandt und die damit einfallende Intensität berechnet;
  - der TransmittedRay wird ausgesandt und die damit einfallende Intensität berechnet;
  - der ambiente Lichtanteil wird berechnet.

  Alle Intensitäten werden durch Superposition überlagert.
- Dies wird für alle EyeRays wiederholt.

Das Aussenden der ReflectedRays und der TransmittedRays ergibt ein rekursives Verfahren. Die Tiefe der Rekursion wird durch eine maximale Rekursionstiefe (RecDepth) begrenzt. Der Algorithmus liefert also einen Strahlbaum:

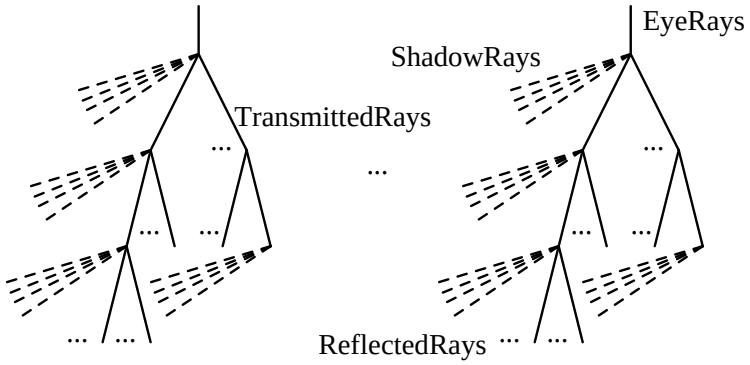

Abbildung: Strahlbaum

Ändert sich die Kameraposition, d.h. der Punkt von dem aus die EyeRays erzeugt werden, oder bewegt sich ein Objekt, dann muß eventuell der gesamte Strahlbaum neu aufgebaut werden.

Als Anzahl der Strahlen, die bei der Berechnung eines Bildes anfallen, erhält man also:

$$r = \#\text{EyeRays} * (2^{\text{RecDepth}} - 1) * (\#\text{ShawdowRays} + 1)$$

Für jeden dieser Strahlen muß der Schnittpunkttest durchgeführt werden. Implementiert man dies auf sehr naive Art und Weise, erfordert dies z.B. ein lineares Suchen über alle Objekte. Die Zeit zur Berechnung eines Schnittpunktes bezeichnen wir im folgenden mit t_i. Will man weiterhin nicht nur ein Einzelbild, sondern eine ganze Sequenz von Bildern erzeugen, vervielfacht sich der Aufwand mit der Länge der Sequenz s. Die Laufzeit beträgt somit (in Einheiten eines Schnittpunkttests) ohne Berücksichtigung der Berechnungen des Beleuchtungsmodells und anderer Vorbereitungs– und Nachbereitungszeiten:

$$T = s * r * t_i$$

Setzt man folgende, durchaus übliche Größen ein: $\#\text{EyeRays} = 512 \times 512$, $\text{RecDepth} = 8$, $\#\text{ShawdowRays} = 5$ erhält man:

$$T \approx s * t_i * 400 * 10^6$$

Nun setzen Beschleunigungsverfahren an allen Faktoren dieser Formel an, d.h. die Anzahl der zu untersuchenden Strahlen wird reduziert und das Auffinden sowie die Berechnung der Schnittpunkte wird beschleunigt.

3.2 Der Algorithmus

RayTracing kann man als rekursiven Algorithmus z.B. wie folgt implementieren. Dabei wird vereinfachend nur von Intensität gesprochen. Wie dies in darstellbare Werte umgesetzt werden kann, bzw. wie Wellenlängenabhängigkeiten einbezogen werden können, iwrd in späteren Abschnitten erläutert.

```
RayTrace
begin
    for all EyeRays do
        Intensity = Trace(EyeRay,0);
    od
end
```

Die Prozedur *RayTrace* berechnet für alle ausgesandten EyeRays die Intensitäten. Dabei wird je Strahl die folgende Funktion *Trace* aufgerufen. Die Berechnung der eigentlichen Intensitäten ist in der Funktion *LightModel* versteckt, wobei weiterhin angnommen wird, daß einfache Superposition gilt.

```
Trace(Ray,RecDepth)
begin
    Intensity=Nothing;
    if NextHitPoint found then
        for all ShadowRays at NextHitPoint do
            Intensity+ =LightModel(NextHitPoint,ShadowRay);
        od
        if RecDepth<MaxDepth then
            Intensity+ =Trace(ReflectedRay,RecDepth+1);
            Intensity+ =Trace(TransmittedRay,RecDepth+1);
        fi
```

```
            Intensity+ =LightModel(NextHitPoint,AmbientLight);
        else Intensity=BackGround
        fi
        return Intensity;
    end
```

Das Berechnen der TransmittedRays und ReflectedRays ist bereits als Übungsaufgabe gestellt worden.

Beim Berechnen der ShadowRays muß man beachten, daß bei transparenten Objekten auch durch das Material beleuchtet werden kann, d.h. man muß den Winkel zwischen ShadowRay und Normalenvektor in die Betrachtung einbeziehen.

RayTracing ist ein Abtastvorgang mit den damit verbundenen Problemen. Das wesentliche Problem liegt im Aliasing (Treppeneffekte). Ein Vorschlag ist *distributed RayTracing*, d.h. eine geringfügige, zufällige Störung der Strahlen abweichend von ihrem idealen, strahlenoptischen Verlauf. Hierzu später mehr in Übungen.

Die Beschleunigungsverfahren greifen an allen Faktoren der oben vorgestellten Formel, d.h. einerseits an der Anzahl der zu verfolgenden Strahlen, andererseits aber auch bei jedem einzelnen Strahl an dessen Berechnungsaufwand, insbesondere dem Auffinden des nächstgelegenen Schnittpunktes. Zudem können parallele Verfahren beim RayTracing leicht eingesetzt werden.

3.3 Beschleunigungsverfahren durch Strahlreduzierung

Die Anzahl der Strahlen kann durch Verringerung der Rekursionstiefe, spezielle Behandlung der ShadowRays, sowie Undersampling–Techniken reduziert werden.

EyeRays brauchen prinzipiell nicht getracet zu werden, da man auf einfachere Algorithmen (z.B. Scanline–Algorithmen) zurückgreifen kann. Ist für ein Pixel das zu sehende Objekt gefunden, kann der Schnittpunkt und Normalenvektor berechnet werden. Ohne Verspiegelung oder Transparenz endet auch dann bereits das Tracen.

Bemerkung: Man beachte allerdings, daß schnelle Verfahren — wie eben der Scanline–Algorithmus — nicht unbedingt mit allen Basisobjekten gut zurechtkommen. Eventuell ist es notwendig alle Basisobjekte in Polygone zu zerlegen, was jedoch dann die Ränder krummliniger Objekte wiederum verschlechtert. Hier muß man wohl auch einen Kompromiß zwischen Qualität und Geschwindigkeit machen.

3.3.1 Rekursionstiefe

Die Rekursionstiefe entscheidet über die sichtbaren Mehrfachspiegelungen, d.h. will man hier eine gewisse Anzahl erreichen, muß auch eine gewisse Rekursionstiefe zugelassen werden. Es zeigt sich jedoch, daß je nach Materialeigenschaften die Intensität, die ein Strahl transportiert, abnimmt. Als ein weiteres Abbruchkriterium kann nun eine untere Schranke für diese Intensität eingfügt werden. Je nach Beleuchtungsmodell muß diese Schranke anders gewählt werden, allerdings kann man im einfachsten Modell das Produkt der materialabhängigen Abschwächungsfaktoren auf dem Pfad im Strahlbaum nehmen.

3.3.2 ShadowRay–Erzeugung

Prinzipiell kann jede Lichtquelle einen Einfluß auf den darzustellenden Punkt haben, somit läßt sich im allgemeinen nichts bezüglich der Anzahl erreichen, allerdings können besondere Eigenschaften der Lichtquelle ausgenutzt werden, so daß die Schnittpunktberechnung für einen ShadowRay stark vereinfacht wird.

- Begrenzt strahlende Lichtquellen erfordern keinen ShadowRay, wenn der NextHitPoint außerhalb der Reichweite der Lichtquelle liegt.
- Ausgeweitet heißt dies, daß geometrisch begrenzte Lichtquellen (siehe Kapitel 2) oft das Aussenden eines ShadowRays unnötig machen.
- Da es genügt ein Objekt zu finden, das bezüglich einer Lichtquelle Schatten wirft, kann man sich eine Liste (kleiner Cache) der zuletzt schattenwerfenden Objekten halten. Man testet zuerst die Elemente dieser Liste, bevor man einen eigentlichen ShadowRay aussendet. (Dies erfordert allerdings eine bestimmte Auswertungsreihenfolge der Schnittpunktberechnungen, so daß im gleichen Schatten liegende Punkten möglichst zeitlich nah ausgewertet werden.)
- Man erzeugt um eine Lichtquelle eine Schattenkarte (ShadowMap), in die die von dieser Lichtquelle beleuchteten Objekte eingetragen werden, da nur beleuchtete Objekte Schatten werfen können. Der Aufbau dieser Karte erfolgt durch eine Diskretisierung einer umgebenden Fläche (Würfel).

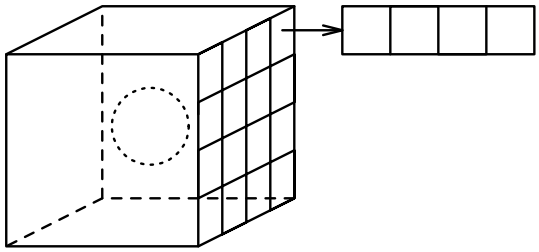

Abbildung: ShadowMap

Die Schattenkarte kann durch einen einfachen präprozessing Lauf je Lichtquelle berechnet werden. Liegt nun das Objekt des NextHitPoint nicht in der Liste des zugehörigen Bereiches, so liegt es im Schatten. Im anderen Fall müssen die in der Liste vorkommenden Objekte als potentielle Schattenspender untersucht werden.

Der Speicherplatz ist maximal

$$O(\#\text{Diskretisierung} * \#\text{Lichtquellen} * \#\text{Objekte})$$

allerdings ist dies in der Praxis oft wesentlich geringer, da nicht jede Lichtquelle jedes Objekt beleuchtet und wenn auch nur in einem kleinen Raumwinkel.

3.3.3 Anzahl der EyeRays

Die Methode des Undersampling reduziert die Anzahl der ausgesendeten EyeRays. Dies führt jedoch dazu, daß nicht mehr alle Bildpunkte exakt berechnet werden.

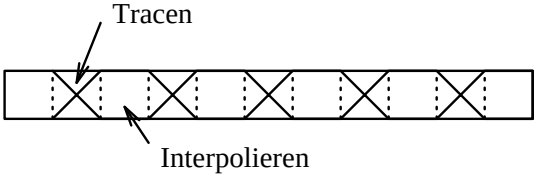

Abbildung: Undersampling

In der einfachsten Version wird der Zwischenpunkt einfach durch Interpolation der Farbwerte der beiden Randpunkte berechnet. Als erste Verbesserung kann untersucht werden, ob beide Randpunkte mit dem EyeRay das selbe Objekt treffen. Die Interpolation wird nur dann vorgenommen, wenn

dies der Fall ist, sonst wird der Bildpunkt normal berechnet. Als weitere Verbesserung kann der Mittelpunkt auf jeden Fall korrekt berechnet werden, allerdings erübrigt sich ein Tracen, falls die Strahlen der Randpunkte das gleiche Objekt treffen, man nimmt einfach an, es wird das gleiche Objekt getroffen. Die Interpolation findet also erst auf zweiter Rekursionsstufe statt. Statt das Verfahren nur in der Bildzeile anzuwenden, ist es auch möglich, dies über Zeilen hinweg durch versetzte Anordnung der Abtastpunkte zu implementieren.

Undersampling birgt eine erhöhte Gefahr des Aliasing. Oft werden Bilder mit Supersampling, d.h. einer höheren Auflösung, berechnet und anschließend auf die eigentliche Auflösung reduziert. Da durch die höhere Auflösung mehr Information vorhanden ist (höhere Abtastfrequenz), können Alias–Effekte reduziert werden. Man wendet also zweckmäßigerweise an Kanten oder anderen Übergängen Supersamling an und an einheitlichen Bildausschnitten Undersampling.

3.3.4 Einfache Parallelisierung

Beim RayTracing ist die Berechnung der einzelnen Bildpunkte relativ unabhängig voneinander. Man kann deshalb gute Beschleunigungen durch einfaches Parallelisieren der EyeRays und damit der Strahlbäume erreichen. Nutzt man jedoch Caching–Strategien oder verwendet man oben erwähnte Undersampling– oder Oversampling–Techniken, ist nicht jede Aufteilung gleich gut. Ferner muß load balancing berücksichtigt werden.

Ein wesentlicher Punkt bei dieser Art der Parallelisierung besteht darin, daß jedem Prozessor die Information über die gesamte Szene zur Verfügung stehen muß. Dies erreicht man entweder dadurch, daß jeder Prozessor die komplette Beschreibung erhält, oder dadurch, daß man eine shared memory–Struktur (zumindest zum Lesen) implementiert. Ist die Szenebeschreibung sehr groß oder dynamisch — man möchte z.B. eine Sequenz von Bildern mit sich bewegenden Objekten erzeugen — dann steigt der notwendige Kommunikationsaufwand in einem verteilten System schnell an.

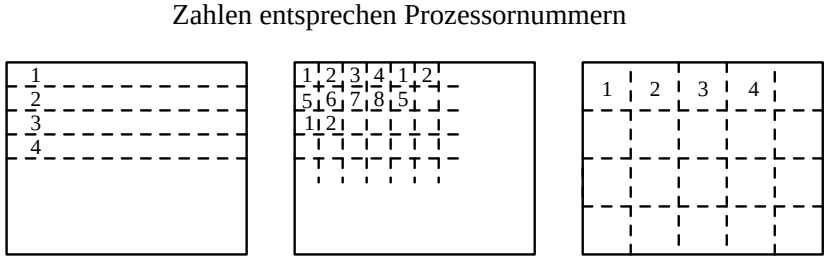

Abbildung: Bildaufteilung bei Parallelisierung

Als einfache Aufteilungsmethoden kommen zeilen– oder spaltenweise Aufteilung, pixelweise feste Zuordnung oder blockweise Zuordnung in Frage (siehe Abbildung). Zudem können Warteschlangen implementiert werden, die den Lastausgleich verbessern. Meiner Meinung ist blockweise Aufteilung mit Warteschlange am besten, da sie auch sehr gut Caching–Strategien ermöglicht.

3.4 Beschleunigungsverfahren bei der Schnittpunktberechnung

Die naive Methode der Schnittpunktsuche führt eine lineare Suche auf allen Objekten der Szene durch, um den nächstgelegenen Schnittpunkt zu finden. Dies kann durch verschiedene Verfahren im Mittel, d.h. für eine Menge von Strahlen, beschleunigt werden.

3.4.1 Einführung von BoundingVolumes

Man kann den Schnittpunkttest in zwei Phasen unterteilen:

- wird überhaupt getroffen und
- wie ist der exakte Schnittpunkt.

Der erste Punkt muß nicht unbedingt exakt erfolgen, man muß lediglich sicher stellen, daß, wenn der Test kein Treffer sagt, dies auch stimmt. Die zweite Phase wird dann eventuell zu oft durchlaufen, aber doch seltener als beim einfachen Verfahren.

Man erreicht dies durch die Einführung von BoundingVolumes, d.h. einfachen geometrischen Körpern, die das eigentliche Objekt möglichst gut umschließen. Man wählt z.B. Kugeln oder achsenparallele Quader.

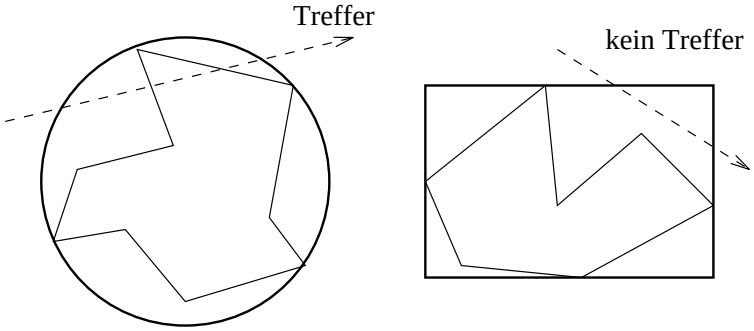

Abbildung: BoundingVolumes

Übungen

1. Sei eine Kugel als BoundingVolume gegeben. Überlegen Sie sich einen Test, der schnell entscheidet, ob die Kugel von einem Strahl getroffen wird oder nicht. (Hinweis: der tatsächliche Schnittpunkt ist anfangs nicht von Interesse)
2. Sei ein achsenparalleler Quader als BoundingVolume gegeben. Überlegen Sie sich einen Test, der schnell entscheidet, ob der Quader von einem Strahl getroffen wird oder nicht. (Hinweis: der tatsächliche Schnittpunkt ist anfangs nicht von Interesse)

Um nun zu entscheiden welches BoundingVolume man benutzt, eignet sich die Definition der Paßgenauigkeit:

$$p = \frac{V(\text{Originalobjekt})}{V(\text{BoundingVolume})}$$

Sei *o* ein Objekt, v_o ein entsprechendes BoundingVolume. Ferner sei

n	die Anzahl der Teststrahlen, die das Volumen v_o bei der Darstellung treffen
$B(v_o)$	die Testkosten für einen Strahl mit v_o
$m(v_o)$	die Anzahl der Treffer von *o* in v_o und
$I(o)$	die Schnittkosten für einen Strahl mit *o*

dann lassen sich die Gesamtkosten berechnen mit

$$K(o, v_o) = nB(v_o) + m(v_o)I(o)$$

$m(v_o)$ läßt sich mithilfe von p abschätzen. Das Ziel ist es nun K durch geeignete Wahl von v_o zu minimieren.

3.4.2 Nutzen von hierarchischen BoundingVolumes

Trotz der Beschleunigung durch den schnellen Test, ob es sehr wahrscheinlich ist, daß das Objekt getroffen wird, bleibt die Zeit für die Suche nach einem Schnittpunkt linear in der Anzahl der Objekte. Ein Verbesserung — zumindest wenn man eine Menge von Strahlen betrachtet — erhält man durch hierarchische BoundingVolumes, indem man bereits vorhandene BoundingVolumes wiederum zu größeren BoundingVolumes zusammenfaßt.

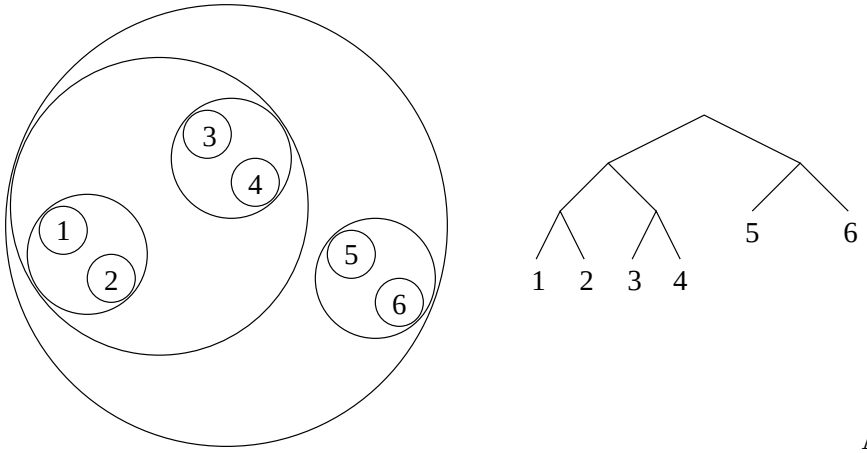

Abbildung: hierarchische BoundingVolumes

Das Erstellen dieser Hierarchie kann einerseits

- durch den Benutzer, also in der Szenendefinition, als auch
- automatisch erfolgen.

Quader eignen sich besser als Kugeln zum Aufbau hierarchischer BoundingVolumes, da im Mittel bei der Vereinigung zweier Quader das Volumen des resultierenden Quaders nicht so sehr ansteigt, wie es bei Kugeln der Fall ist.

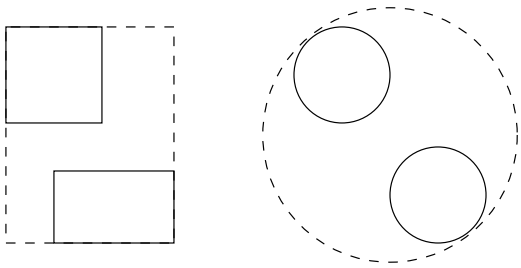

Abbildung: Vergleich: Kugel oder Quader als hierarchisches BoundingVolume

Übungen

1. Seien die BoundingVolumes als Kugeln um die Objekte einer Szene gegeben. Überlegen Sie sich ein automatisches Verfahren, daß die Kugeln hierarchisch zu immer größeren BoundingVolumes zusammenfaßt.
2. Gegeben sei ein binärer Baum, der die Hierarchie der BoundingVolumes repräsentiert. Geben Sie einen Algorithmus an, der in dieser Darstellung den nächsten Schnittpunkt eines beliebigen Strahles (Gegeben durch Aufpunkt und Richtung) findet.

Das die Suchzeit pro Strahl weiterhin linear in der Anzahl der Objekte sein kann, sieht man in folgender Abbildung:

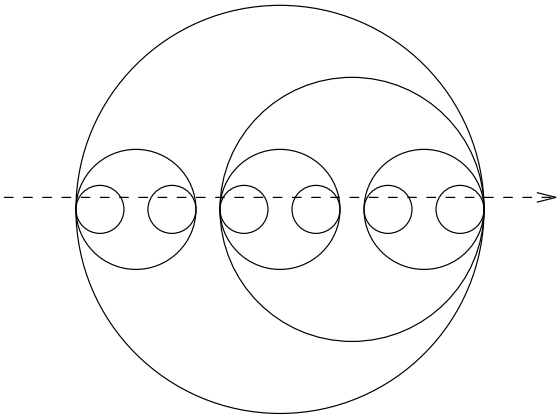

Abbildung: alle BoundingVolumes werden geschnitten

Die Laufzeit bleibt also im schlechtesten Fall linear, im Mittel ist sie allerdings wesentlich geringer. Da die Verwendung der BoundingVolume auf jeden Fall eine Beschleunigung mit sich bringt, gehen wir in den folgenden Raumunterteilungsverfahren stets davon aus, daß mit BoundingVolumes gearbeitet wird (allerdings nicht notwendigerweise hierarchisch).

3.4.3 Raumunterteilungsverfahren

Unter einer Raumunterteilung verstehen wir die Unterteilung des dreidimensionalen Raumes, in dem sich die Objekte befinden, in kleinere Unterräume. Sind die Unterräume achsenparallele Quader, dann heißen die Unterräume Voxel. Zwei Voxel heißen ähnlich, wenn sie durch gleiche Skalierung der drei Achsen ineinander übergeführt werden können. Sind alle Voxel, die bei einer Aufteilung entstehen, ähnlich, sprechen wir von einer uniformen Unterteilung, im andern Fall von einer nicht–uniformen Unterteilung. Handelt es sich um eine mehrstufige Unterteilung, sprechen wir von einer hierarchischen Unterteilung, sonst von einer nicht–hierarchischen Unterteilung.

Damit ist klar, daß es verschiedene Raumunterteilungsverfahren gibt. Ziel jeder Unterteilung ist es, den Suchraum bei der Schnittpunktberechnung einzuschränken, indem jedem Unterraum nur die in im liegenden Objekte zugeordnet werden (hierzu sind eventuell Kopien notwendig). Durchstreift ein Strahl den Unterraum, brauchen nur diese wenigen Objekte als Kandidaten für einen Schnittpunkt untersucht zu werden.

Um zu verhindern, daß der exakte Schnittpunkt eines Strahls mit einem Objekt mehrfach berechnet wird, wenn das Objekt mehrere Voxel schneidet, merkt man sich Strahlnummern und die zuletzt berechneten Schnittpunkte. Vor einer erneuten Schnittpunktberechnung testet man nun zuerst die Strahlnummer mit dem aktuellen Strahl und spart sich so eine erneute Berechnung. Weiterhin muß man darauf achten, daß der Schnittpunkt im aktuellen Voxel liegt. Dies ist eine notwendige Bedingung für den nächsten Schnittpunkt.

Folgende Abbildung klassifiziert mögliche Raumunterteilungsverfahren, die im Anschluß genauer besprochen werden:

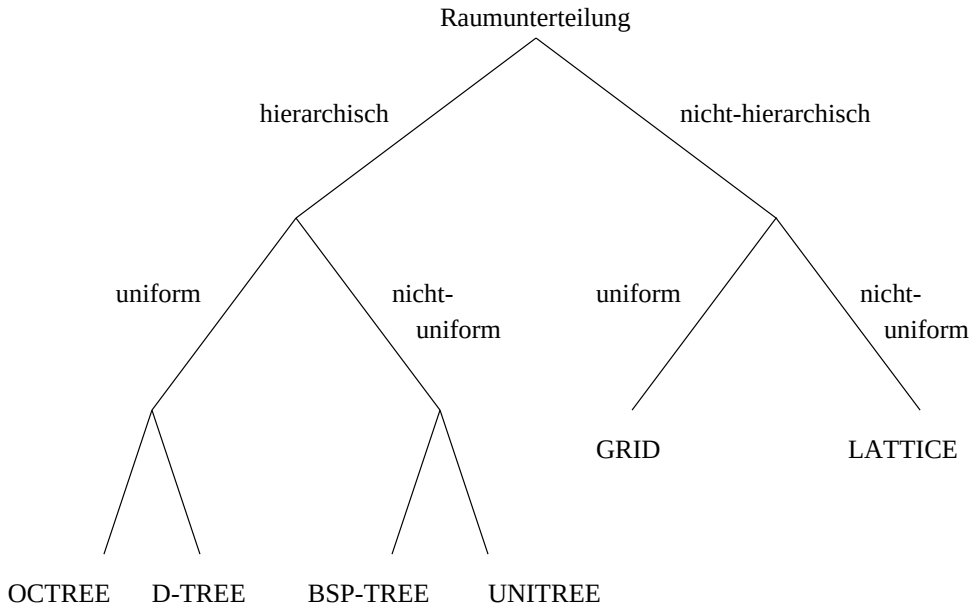

Abbildung: Raumunterteilungsverfahren

Bemerkung: Obige Definitionen entsprechend nicht den in der Literatur sonst üblichen Definitionen. Dort wird meist das, was hier hierarchisch ist, als nicht–uniform und das, was hier nicht–hierarchisch ist, als uniform bezeichnet.

Die Beschreibung enthält stets eine grobe Abschätzung des Aufwandes, der bei der Schnittpunktberechnung notwendig ist. Dabei wird immer von einer regelmäßigen oder zufälligen Anordnung vieler relativ kleiner Objekte ausgegangen. Zufällig soll hier nicht formal eingeführt werden, sondern damit soll lediglich gesagt werden, daß man erwartet, in einem beliebigen, nicht zu kleinen, aber konstantem Raumvolumen im wesentlichen die gleiche Anzahl von Objekten vorzufinden, die diesen Unterraum schneiden.

Die Anzahl der Objekte der Szene sei n. Müssen durch die Raumunterteilung Objektinformationen kopiert werden, so bezeichne k den Faktor, um den sich die Objektinformation erhöht. Als Beispiel nehme man an, daß jedes Voxel einen Zeiger auf jedes Objekt enthält, welches das Voxel schneidet. Müssen 10 % der Zeiger dupliziert werden, ist $k = 1.1$. Desweiteren sei d die Anzahl der Unterräume, die ein Strahl im Mittel trifft, wenn er die Szene durchläuft.

GRID Der Raum, den die Szene einnimmt, wird durch ein Grid in gleich große Voxel eingeteilt. Übliche Einteilungen verwenden bis zu $10 \times 10 \times 10$ Voxel, d.h. es entstehen 1000 Unterräume. Jedem Voxel werden die dieses Voxel schneidenden Objekte (deren BoundingVolumes) zugeordnet. Schneidet ein Objekt mehr als ein Voxel, muß es also in mehreren Voxeln gehalten werden (zumindest ein Verweis auf das Objekt).

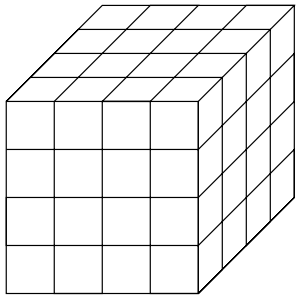
Abbildung: GRID

Die Schnittpunktsuche beginnt in dem Voxel, das den Aufpunkt des Strahles enthält. Trifft er kein Objekt innerhalb des Voxels, wird das nächste Voxel mithilfe eines modifizierten BRESENHAM–Algorithmus bestimmt und die Suche dort fortgesetzt.

Grobe Abschätzung des Berechnungsaufwandes:

Gehen wir von c^3 vielen Voxeln aus. In jedem Voxel liegen dann $k * n/c^3$ viele Objekte. Für einen Strahl sind im Mittel d Voxel zu testen. Bei einer Durchquerung der Szene können höchstens $3c$ viele Voxel geschnitten werden. Für sehr dicht besetzte Szenen kann man davon ausgehen, daß die Anzahl der durchquerten Voxel unabhängig von c ist; die Anzahl der geschnittenen Voxel kann dann sogar als Konstante auftreten.

Als Berechnungszeit für die Schnittpunktbestimmung ergibt sich also zu

$$O(dk\frac{n}{c^3})$$

und somit im schlechtesten Fall (alle Objekte schneiden alle Voxel) zu

$$O(3cn)$$

Die Methode lohnt sich also nur dann, wenn gilt:

$$dk < c^3$$

Der Aufwand zum Finden der Voxel, die der Strahl schneidet, ist $O(d)$ somit höchstens $O(c)$.

Im Falle $c = 8$, $k = 1.1$ und $d = 3$ ergibt sich also eine Reduzierung auf das 0.0065–fache der Zeit der naiven linearen Testmethode. Hierbei wurden weitere Konstanten vernachlässigt. Allerdings sind diese nicht wesentlich größer als bei der linearen Suche, da man genau diese innerhalb der Voxel auch benutzt und das Finden der Voxel mithilfe des BRESENHAM–Algorithmus sehr schnell geht.

Der Speicherplatz erhöht sich linear, da für jedes Objekt ein Verweis in mindestens einem Voxel gehalten werden muß. Sind allerdings Kopien notwendig, erhöht sich der Platz auf $O(kn)$, was im schlechtesten Fall zu $O(c^3n)$ führen kann.

LATTICE Hier wird der Raum analog zum GRID in quaderförmige Unterräume eingeteilt. Allerdings findet diese Unterteilung nicht an festen Stellen statt, sondern die Objektgrenzen werden berücksichtig. Dies führt dazu, daß man durch eine günstige Plazierung der Schnittebenen weniger Objekte schneidet und somit auch weniger Objektinformation kopieren muß, d.h. k ist geringer als beim GRID. Die Anzahl der im Mittel von einem Strahl geschnittenen Quader ist ebenfalls nicht höher als beim GRID.

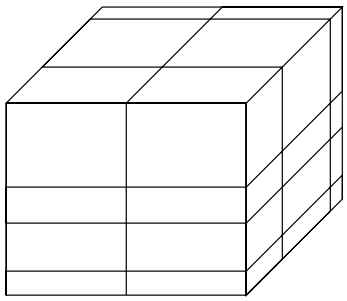

Abbildung: LATTICE

Die Laufzeitabschätzung ist ähnlich der des GRID, nun jedoch mit unterschiedlichen Einteilungen in den drei Raumrichtungen. Man erhält also

$$O(dk\frac{n}{c_x c_y c_z})$$

und somit im schlechtesten Fall (alle Objekte schneiden alle Voxel)

$$O((c_x + c_y + c_z)n)$$

Die Methode lohnt sich also nur dann, wenn gilt:

$$dk < c_x c_y c_z$$

Wie gesagt, erwartet man ein kleineres k als beim GRID und zudem lassen sich auch in nicht regelmäßigen, Cluster–artigen Szenen gute Aufteilungen des Raumes finden.

Für den BRESENHAM–Algorithmus müssen nun auch die Intervallängen der Einteilungen in X–, Y– und Z–Richtung gespeichert werden, was den Platzbedarf um $O(c_x c_y c_z)$ erhöht (was bei vielen Objekten vernachlässigt werden kann). Ansonsten steigt der Platzbedarf wie beim GRID.

OCTREE Der Raum wird durch wiederholte Halbierung der Kanten des die Szene umgebenden achsenparallelen Quaders in immer kleinere Quader unterteilt. Wird diese Unterteilung balanciert ausgeführt, haben alle Voxel die gleiche Größe (weshalb ich es — im Gegensatz zur Literatur — als uniforme Raumunterteilung bezeichne). Normalerweise wird weitere Unterteilung eines Unterraumes abgebrochen, sobald die Anzahl der Objekte innerhalb eines Voxels eine bestimmte Untergrenze unterschreitet, d.h. hinreichend leere Voxel werden nicht weiter unterteilt.

Die Abbildung zeigt einen QUADTREE (das zweidimensionale Analogon) mit der Abbruchbedingung, das keine weitere Unterteilung im Falle von zwei Objekten im Unterraum stattfindet.

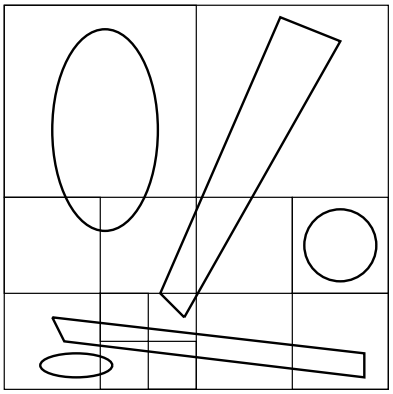

Abbildung: OCTREE

Wiederum werden jedem Voxel als Blatt im Baum seine Objekte mitgeteilt. Dies bedeutet, daß ebenfalls Information kopiert werden muß, wenn ein Objekt mehrere Voxel schneidet.

Wenn sich Objekte überlappen können, muß ein weiteres Kriterium als Rekursionsabbruch beim Aufbau des OCTREE berücksichtigt werden, da die vorgegebene minimale Anzahl nicht notwendigerweise erreicht werden muß: es kann vorkommen, daß eine weitere Unterteilung keine Reduktion der Objektanzahl in einem Voxel mit sich bringt.

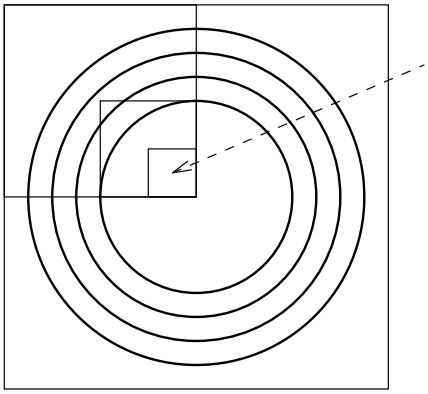

Abbildung: OCTREE und sich überlappende Objekte

Als mögliche Kriterien können z.B. genutzt werden:

- Durch die Unterteilung sinkt die Anzahl der Objekte in einem Unterraum nicht auf mindestens einen vorgebenen Prozentsatz des Oberraumes.

$$\#\text{Obj}_{\text{Unterraum}} \geq s * \#\text{Obj}_{\text{Oberraum}} \quad \text{mit} \quad 0 < s < 1$$

- Durch die Unterteilung steigt die Summe aller Objekte in den Unterräumen auf mehr als ein vorgegebenes Vielfaches der Anzahl im Oberraum.

$$\sum \#\text{Obj}_{\text{Unterraum}} \geq s * \#\text{Obj}_{\text{Oberraum}} \quad \text{mit} \quad 1 < s < 8$$

Die Schnittpunktberechnung startet wieder in dem Voxel, das den Aufpunkt des Strahles enthält. Dieses wird durch einen top–down–Lauf durch den Baum gefunden. Das Auffinden des nächsten Voxel, das der Strahl beim Verlassen des vorhergehenden betritt, wird wie folgt gefunden:

1. Bestimmen eines Punktes, der innerhalb dieses Voxels liegt und
2. Suchen des Voxel mithilfe dieses Punktes

Eine mögliche Implementierung dieser beiden Schritte arbeitet wie folgt:

1. Man bestimmt den Schnittpunkt des Strahls mit der Oberfläche des aktuellen Voxel oder auch nur die Seitenfläche, die getroffen wird. Nun wählt man einen Punkt außerhalb des aktuellen Voxel in der dominanten Strahlrichtung, der die halbe, minimale Voxelausdehnung entfernt liegt. Dies kann entweder vom gefundenen Wandschnittpunkt (Sonderfall der Kanten muß man berücksichtigen) oder vom Zentrum der Seitenfläche aus erfolgen. Dieser Punkt liegt nun mit Sicherheit im nächsten getroffenen Voxel.

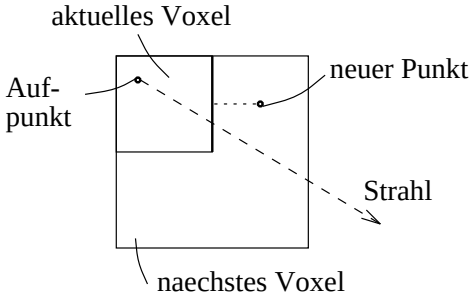

Abbildung: Finden des nächsten geschnittenen Voxel im OCTREE

2. Das entsprechende Voxel findet man nun wieder durch einen top–down–Lauf durch den Baum. Ist d sehr klein, d.h. die Strahlen legen im Mittel nur kleine Wege in der Szene zurück, lohnt es sich, erst im Baum ein Stück noch oben zu suchen, bis man das Voxel gefunden hat, das sowohl den Aufpunkt als auch den neuen Punkt enthält, und dann abwärts dasjenige Voxel sucht, das nur den neuen Punkt enthält.

Grobe Abschätzung des Berechnungsaufwandes:

Sei c die Baumtiefe des OCTREE. Da die Objekte regelmäßig/zufällig verteilt sind, ist der Baum im wesentlichen balanciert. Es gibt somit 2^{3c} Voxel als Blätter, die Objektinformation enthalten müssen. Sei k der Faktor auf den sich die Objektanzahl bei der Unterteilung eines Voxel erhöht. Für einen Baum der Tiefe c erhöht sich die Objektanzahl auf

$$\begin{aligned} n + (k-1)n + 2(k-1)n + \ldots + 2^{c-1}(k-1)n \\ = ((2^c - 1)(k-1) + 1)n \end{aligned}$$

wenn man annimmt, daß jede Schnittebene in der Szene die gleiche Anzahl von Objekten schneidet. In jedem Voxel als Blatt liegen also

$$\frac{((2^c - 1)(k-1) + 1)n}{2^{3c}}$$

Objekte und als Laufzeit ergibt sich

$$O(d\frac{((2^c - 1)(k-1) + 1)n}{2^{3c}})$$

Mit der gleichen Argumentation wie beim GRID werden d Voxel als Blätter von einem Strahl durchlaufen.

Den Faktor für die Tiefe c, der die Anzahl der Objektinformationen angibt, sieht man leicht an folgender Abbildung:

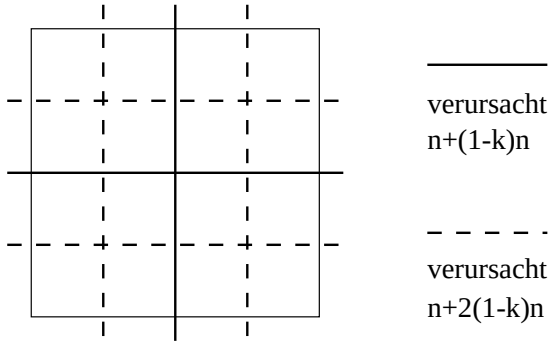

Abbildung: Kopierfaktor beim OCTREE

Für jeden Voxel muß ein top–down–Lauf durch den Baum implementiert werden, was langsamer als ein Auffinden des nächsten Voxel mithilfe des BRESENHAM–Algorithmus ist: man erhält $O(d * c)$, also als Maximum mit $d = 3 * 2^c$ ergibt sich $O(c2^c)$.

Ist die Anzahl der Voxel als Blätter des OCTREE gleich der Anzahl der Voxel im GRID, dann ergeben sich die gleichen Laufzeiten bei der Schnittpunktsuche. Nur im Auffinden des nächsten geschnittenen Voxels verliert man mehr Zeit beim OCTREE als beim GRID. Der Faktor k des GRID entspricht dem Faktor $((2^c - 1)(k - 1) + 1)$ des OCTREE, da die Blätter des OCTREE den Raum analog zu einem GRID aufteilen.

Somit ist klar, daß der OCTREE dem GRID nicht überlegen ist, wenn es sich um regelmässige/zufällige Szenen handelt. Der OCTREE gewinnt erst, wenn es sich um unregelmässige, clusterartige Objektansammlungen handelt und man durch den OCTREE schnell große Raumbereiche durchdringen kann. In diesem Fall ist es möglich — durch nicht balancierte Bäume — die Zahl der in einem Blatt gespeicherten Objekte für alle Voxel als Blätter eher gleich zu halten oder auch sehr klein.

BSP–TREE Der binary space partition tree entspricht dem OCTREE, allerdings wird die Unterteilung der Quader ähnlich wie beim LATTICE anhand von Objektgrenzen durchgeführt. Man versucht durch jede Unterteilung links und rechts der Schnittebene die gleiche Anzahl von Objekten zu erhalten und die Anzahl der zu duplizierenden Objekte gering zu halten. Man unterteilt also nicht notwendigerweise stets abwechselnd in den drei Raumdimensionen.

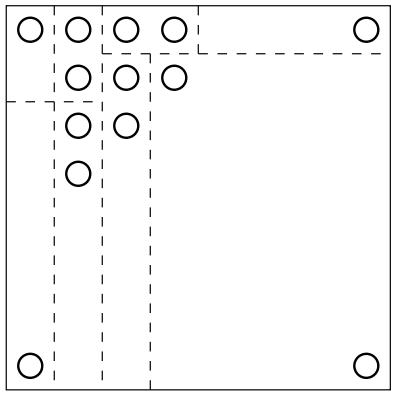

Abbildung: BSP–TREE

Die Schnittpunktsuche erfolgt analog zu dem Verfahren ab, wie es für den OCTREE beschrieben worden ist. Die Laufzeitabschätzung für einen BSP–TREE entspricht bei der vorgegebenen Szenenbeschreibung der des OCTREE, da aufgrund der Objektverteilung die Schnittebenen an die gleiche Stelle gelegt werden.

Ein BSP–TREE hat also auch nur den Vorteil, eine gleichmäßigere Aufteilung der Objekte auf die Voxel als Blätter zu ermöglichen. Dies erweist sich jedoch bei clusterartigen Objektanordnungen als deutlicher Vorteil.

UNITREE In einem Voxel bei GRID oder LATTICE bzw. in einem Voxel als Blatt bei OCTREE oder BSP–TREE bleibt die Anzahl der Objekte beschränkt. Die Frage stellt sich, wie man dort den Schnittpunkt findet, wenn es mehr als ein Objekt enthält. Es stehen folgende Möglichkeiten zur Verfügung:

- lienare Suche,
- hierarchische BoundingVolumes und
- PlaneTraversal.

PlaneTraversal arbeitet wie folgt. Jede Seitenfläche eines BoundingVolumes (wir setzen Quader als BoundingVolumes voraus) definiert eine Ebene. Man erhält also $6 * \#$Objekte viele Ebenen. Jede Ebene kennt ihr Objekt. Die Ebenen sind entsprechend ihrer Koordinate sortiert. Der Strahl durchläuft das Voxel — dabei werden die Ebenen geschnitten — und aktiviert bzw. deaktiviert ein entsprechendes Bit je Koordinate pro BoundingVolume je nachdem, ob das BoundingVolume des Objektes potentiell in dieser Koordinate betreten oder verlassen wird. Sind alle drei Bits aktiviert, wird das BoundingVolume getroffen und ein Schnittpunkttest kann erfolgen. Man kann die Suche abbrechen, sobald eine Ebene getroffen wird, die hinter dem letzten gefundenen Schnittpunkt liegt.

Da dieses Verfahren sehr effizient implementiert werden kann, arbeitet es bis zu einer gewissen Objektanzahl schneller als ein Verfahren mit hierarchischen BoundingVolume.

Überlappen sich BoundingVolume stark, ist es nicht unbedingt sinnvoll ein übergeordnetes BoundingVolume zu konstruieren, da für die meisten Strahlen sowieso beide getestet werden müssen. In diesem Fall erweitert man die hierarchische BoundingVolume–Struktur dadurch, daß man an den Knoten mehr als zwei Söhne zuläßt. Bei der Schnittpunktsuche muß dann jedoch stets die gesamte Liste der Söhne durchsucht werden. Dies kann jedoch wiederum schnell mit PlaneTraversal erfolgen.

Eine einfache Modifizierung des OCTREE, die nicht nur 8 Unterräume pro Unterteilung eines Voxels erzeugt und damit eventuell Objektinformation kopieren muß, besteht darin, alle durch Schnittebenen des OCTREE (oder auch eines BSP–TREE) geschnittenen Objekte in ein eigenes Voxel (Universum) zusammenzufassen. Dabei können also je Unterteilungsschritt bis zu 255 Unterräume entstehen. (In jedem Unterraum sind die Objekte zusammengefaßt, die durch die gleichen Schnittebenen des OCTREE geschnitten werden.)

Man beachte, daß sich nun die entstandenen Voxel durchdringen können, und man somit das oben erläuterte Verfahren mit sich überlappenden BoundingVolumes nutzen muß.

Das in diesem Abschnitt skizzierte Verfahren (UNITREE) mit modifiziertem BSP–TREE und überlappenden BoundingVolumes (Universen) ist in dem am Lehrstuhl Prof. W.J. PAUL implementierten RayTracer genutzt worden.

D–TREE Der direction tree ist ein Unterteilungsverfahren, das man auch unter uniforme Raumunterteilung einordnen kann. Der Raum wird jedoch nicht einfach in Voxel unterteilt, sondern man macht sich die Tatsache zu nutzen, daß ein Strahl nur 5 Freiheitsgrade hat. Er läßt sich nämlich durch seinen Aufpunkt (drei Werte) Polarkoordinaten (zwei Werte) verwenden). Da man geeigneter Weise keine Polarkoordinaten sondern (u, v)–Werte auf einer umgebenden Oberfläche (Würfel) nimmt, ergibt sich als weiterer Freiheitsgrad eine der sechs Hauptachsenrichtungen.

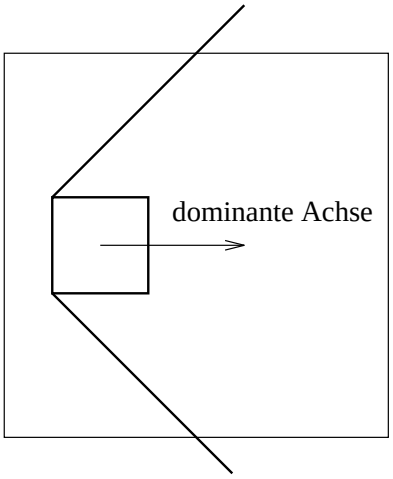

Abbildung: Richtungswürfel

Die Raumunterteilung erfolgt nun rekursiv, abwechselnd in den fünf Dimensionen. Die Hauptachsenrichtung spielt natürlich auch eine Rolle und bewirkt eine Multiplikation mit 6. In den ersten drei Dimensionen entstehen Voxel analog zum OCTREE–Verfahren. In den anderen beiden Dimensionen entstehen Pyramiden. Die Objekte werden wiederum jedem Raumbereich zugeordnet. In den Pyramiden werden die Objekte zusätzlich in der Richtung der Pyramide sortiert, so daß der nächstgelegene Schnittpunkt schneller gefunden werden kann.

Das Verfahren traversiert also den entstehenden Baum und bestimmt jeweils in welchen Raumbereich der Strahl fällt. An einem Blatt wird dann der Schnittpunkt berechnet. Der Speicherplatzverbrauch dieser Methode ist erheblich, da sich die Pyramiden deutlich überlappen. Man kann fast sagen, daß jedes nur durch die ersten drei Raumdimensionen erzeugte Voxel den gesamten Objektraum sieht, d.h. die Anzahl der Blätter eines OCTREE (bezüglich der X–, Y– und Z–Koordinate entsteht gerade ein OCTREE) bestimmt, wie oft die Objektinformationen kopiert werden müssen, da man von einem solchen Voxel, den gesamten Raum sehen kann. Eine Reduzierung erhält man durch einen adaptiven Aufbau der Datenstruktur mit caching–Strategien (lazy evaluation). Einen interessanten Beschleunigungseffekt erhält man auch durch die Begrenzung der Ausdehenung der Pyramiden. Jeder Raumpunkt sieht praktisch nur eine gewisse Distanz weit, d.h. für jeden Punkt können sehr weit entfernte Objekte vernachlässigt werden. Bei den Lichtquellen wurde diese Methode weiter oben bereits vorgestellt: es wird nur das erste getroffene Objekt in einem Raumwinkel gespeichert, da alle dahinterliegenden automatisch in Schatten liegen.

Als grobe Laufzeitanalyse erhält man, wiederum für eine große Anzahl regelmäßig/zufällig verteilter, kleiner Objekte:

Wir nehmen an, daß der Baum balanciert aufgebaut ist. Dann liegen in jeder Pyramide etwa gleich viele Objekte. Sei k wiederum der Faktor, der angibt um wieviel die Anzahl der Objekte vergrößert werden muß, wenn eine Unterteilung in allen fünf Dimensionen erfolgt ist. Wir nehmen der Einfachheit wegen an, daß in jedem Rekursionsschritt die Anzahl der Objektinformationen um den gleichen Faktor k steigt.

Sei c die Tiefe des Baumes, wobei in jeder Dimension unterteilt worden ist. Mit jedem Rekursionsschritt wächst die Anzahl der Blätter also um den Faktor $2^5 = 32$. In einer Pyramide als Blatt liegen also

$$O(\frac{k^c}{2^{5c}}n)$$

Objekte.

Bei der Schnittpunktberechnung wird ein Pfad des Baumes mit Länge $5c$ (wegen der Unterteilung in allen Dimensionen) durchlaufen, allerdings braucht nur ein Blatt untersucht zu werden, da der Strahl ganz in einer Pyramide verläuft. Man erhält also:

$$O(2^{c(\log k - 5)}n)$$

als Laufzeit. Die Bestimmung der Hauptachsenrichtung taucht nur als Konstante in jedem Knoten des Pfades auf. Bei einer Tiefe $c = 3$ (entspricht 32768 Blättern) und einer Überlappung von $k = 8$ beträgt die Beschleunigung 0.0156.

Übungen

1. In der Vorlesung wurde distributed ray tracing angesprochen. Erläutern Sie zu welchen Zwecken man dieses Verfahren für welche Strahlen anwenden kann.

2. Sie wollen wellenlängenabhängiges RayTracing implementieren. Hierzu müssen Sie mehrere Strahlen bezüglich ihrer Wellenlänge verfolgen. Wie integrieren Sie dieses in den herkömmlichen RayTrace–Vorgang, d.h. in das Beleuchtungsmodell sowie die Strahlerzeugung.

3. Erläutern Sie nochmals kurz die Vor– und Nachteile der Beschleunigungsverfahren bei der Schnittpunktberechnung. Kann man leicht die Vorteile der Verfahren miteinander kombinieren ? (Unterscheiden Sie dabei automatische Verfahrensweisen und durch den Benutzer/Modellierer zur Verfügung gestellte Informationen.)

4 Gekrümmte Linien und Oberflächen

Literatur:

- RAUBER, THOMAS; Algorithmen in der Computer Graphik; *erscheint in Springer Verlag, voraus. 1993*
- FOLEY, JAMES; VAN DAM, ANDRIES; Computer Graphics, Principles and Practice; *2nd Edition, Addison Wesley, 1991*
- FARIN GERALD; Curves and Surfaces for Computer Aided Geometric Design; *Academic Press, San Diego, USA, 1990*
- HOSCHEK JOSEF, LASSER DIETER; Grundlagen der geometrischen Datenverarbeitung; *Teubner Verlag, Stuttgart, 1989*

Polygone sind leicht zur Modellierung in der ComputerGraphik einzusetzen. Um jedoch gekrümmte Oberflächen, was die Mehrzahl der Oberflächen in der Natur sind, darzustellen, führt das Nutzen einfacher, eben flacher, Polygone gerade an Kanten und Rändern der Fläche zu Problemen.

- Die weichen, glatten Kanten werden durch die Polygone in abschnittsweis lineare Segmente zerlegt.
- Um dies zu vermeiden, kann man die Approximation der Oberfläche durch eine Steigerung der Polygonanzahl verbessern.
- Wird dies beim Modellieren explizit gemacht, erfordern die Polygone sehr viel Speicherplatz.
- Je nach Darstellungsverfahren (wir konzentrieren uns auf RayTracing) geht eventuell Information über den Zusammenhang der Polygone verloren, was jedoch ausgenutzt werden könnte (um die Schnittpunktberechnung zu beschleunigen).

Man sucht deshalb nach einer Möglichkeit sowohl die Modellierung als auch die Darstellung von gekrümmten Oberflächen zu vereinfachen.

4.1 Überblick

Man kann zur Beschreibung von Oberflächen im wesentlichen drei Methoden unterscheiden:

analytische Beschreibung geometrischer Oberflächen

approximierende Beschreibung durch Kontrollpunkte oder Stützstellen, die Fläche geht NICHT DURCH die Kontrollpunkte

interpolierende Beschreibung durch Kontrollpunkte oder Stützstellen, die Fläche geht DURCH die Kontrollpunkte

Bemerkung: Wir machen im folgenden keine Unterscheidung zwischen Vektoren des Vektorraumes und Punkten des EUKLIDschen Raumes, d.h. wir identifizieren Ortsvektoren, Vektoren und Punkte und unterlassen auch die Kennzeichnung durch Pfeile oder ähnliches. Es sei jedoch darauf hingewiesen, daß dies sehr wohl unterschiedliche Dinge sind und man in gewissen Fällen eine Unterscheidung machen muß.

4.1.1 Analytische Beschreibungen

Als einen Ansatz kann man die Beschreibung der Oberflächen auf wohlbekannte geometrische Körper stützen:

- einfache geometrische Beschreibung der Oberfläche: z.B. als Kugel, Kegel oder Torus;
- Rotationskörper, die zumindest in der Drehrichtung keine Diskontinuitäten aufweisen;
- Kegelschnitte, wie Parabeln, Elipsen und Hyperbeln, oder trigonometrische Beschreibungen, wie Sinuswellen;
- geometrische Beschreibung durch Quadriken, d.h. Oberflächen, die durch $4\times$–Matrizen beschrieben werden, also eine allgemeine analytische Beschreibung, die es erlaubt, Ellisoide, Tori, Paraboloide usw. einheitlich zu beschreiben.

In allen diesen Fällen lassen sich die Schnittpunkte mit einem Strahl entweder analytisch berechnen (z.B. Kugel (siehe Übungen)) oder zumindest mittels iterativer Methoden schnell und beliebig genau annähern.

Der wesentliche Nachteil dieser Beschreibungsmethode sind die eingeschränkten Modellierungsmöglichkeiten. Eine solche Oberfläche verliert nie ihren geometrischen Charakter. Weiterhin ist die Beschreibung stets global, d.h. es wird die Oberfläche eines Körpers definiert. Lokale Änderungen der Oberfläche sind nicht oder nur sehr schwer durchführbar.

Die ersten drei Methoden sind relativ einfach zu handhaben und gehören in den Bereich, den wir in der Einleitung ausgeschlossen haben. (Quadriken machen wir vielleicht noch am Ende).

4.1.2 Approximierende Beschreibung

Man benutzt zur Beschreibung eine Menge von Kontrollpunkten oder Stützstellen und einen Algorithmus, der angibt wie man alle Punkte der Oberfläche aus den Kontrollpunkten berechnen kann. Die Kontrollpunkte selbst gehören nicht unbedingt der Oberfläche an.

Wir werden folgende Typen von Beschreibungen erläutern:

- BÉZIER–Typen
- B–Spline–Typen

Die Beschreibungen erfolgen in Vektornotation, alle Darstellungen sind in zweidimensionaler Form gegeben. Die Algorithmen sind unabhängig von der Dimension. Dreidimensionale Kurven werden durch die Parameterbeschreibung in den drei Dimensionen angegeben, wobei die entsprechenden Polynome durch den gemeinsamen Parameter gekoppelt sind.

4.1.3 Interpolierende Beschreibung

Analog zu den approximierenden Verfahren wird hier eine Menge von Kontrollpunkten und ein Algorithmus dazu verwendet, die Oberfläche zu definieren, allerdings liegen die Kontrollpunkte hier in der Oberfläche.

Wir werden folgende Typen von Beschreibungen erläutern:

- LAGRANGE–Typen
- B–Spline–Typen

4.2 BÉZIER–Kurven

Aus den 60er Jahren stammt die Beschreibung gekrümmter Kurven durch BÉZIER–Kurven, die unabhängig voneinander BÉZIER und DE CASTELJAU entwickelt hatten.

Seien $b_0, \ldots, b_n$ die Kontrollpunkte der Kurve. Eine BÉZIER–Kurve hat die folgenden Eigenschaften:

1. Sie verläuft durch die Endpunkte b_0 und b_n.
2. Sie liegt vollständig in der konvexen Hülle der Kontrollpunkte.
3. Die Steigung der Kurve in den Endpunkten ist identisch mit der Steigung durch die Geraden, die durch die Endpunkte und deren Nachbarpunkte definiert werden, also (b_0, b_1) bzw. (b_{n-1}, b_n).
4. Die Berechnung ist numerisch sehr stabil, d.h. es findet keine Fehlerakkumulation statt.
5. Sie können durch einen rekursiven Algorithmus (den DE CASTELJAU–Algorithmus) oder
6. durch Polynome (die BERNSTEIN–Polynome) beschrieben werden.
7. Alle Kontrollpunkte sind notwendig, um einen Punkt auf der BÉZIER–Kurve zu berechnen (Ausnahme: die beiden Randpunkte).
8. Sie sind invariant gegenüber affinen Transformationen.
9. Sie sind nicht invariant gegenüber Projektionen.

4.2.1 Der DE CASTELJAU–Algorithmus

Der Algorithmus von DE CASTELJAU erzeugt durch eine Rekursion für einen Wert eines Parameter t im Intervall $[0, 1]$ einen Punkt der Kurve, d.h. jedem Wert für t entspricht genau ein Punkt der Kurve. Man kann also durch eine Auswahl von Punkten im Intervall $[0, 1]$ eine entsprechende Anzahl von Punkten der Kurve bestimmen und diese dann z.B. durch Geradensegmente miteinander verbinden (Eigenschaft 5)).

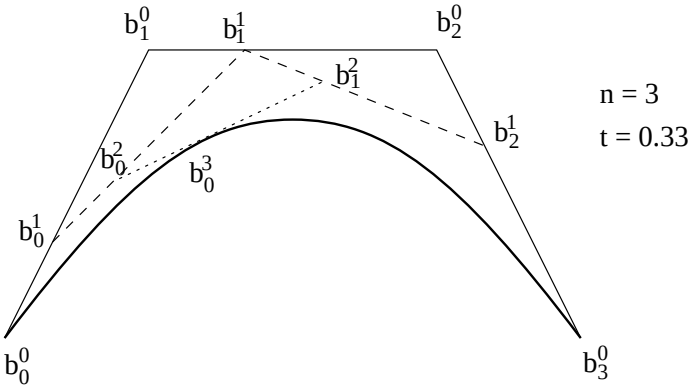

Abbildung: Algorithmus von DE CASTELJAU

Seien $b_0, \ldots, b_n$ die Kontrollpunkte der Kurve. Durch einen weiteren Index bezeichnen wir die Tiefe der Rekursion.

Wir identifizieren $b_0, \ldots, b_n$ mit $b_0^0, \ldots, b_n^0$ und berechnen rekursiv für $1 \leq r \leq n$ und $0 \leq i \leq n - r$:

$$\begin{aligned} b_i^r &= b_i^{r-1} + t * (b_{i+1}^{r-1} - b_i^{r-1}) \\ &= (1-t) * b_i^{r-1} + t * b_{i+1}^{r-1} \end{aligned}$$

wie dies obige Abbildung verdeutlicht. Die Rekursionstiefe ergibt sich also als Anzahl der Kontrollpunkte minus 1 (Da wir jedoch bei 0 anfangen zu zählen, ist gerade die Rekursionstiefe gleich n). Man bezeichnet n auch als Grad der BÉZIER–Kurve.

Zur Auswertung der Rekursion genügt ein Feld der Lange $(n+1)$, da die Einträge geeignet überschrieben werden können.

Wie man sieht, wird nie ein Punkt außerhalb der konvexen Hülle konstruiert, was die Benutzung von BoundingVolumes erleichtert (Eigenschaft 2)). Ferner geht die Kurve durch die beiden Endpunkte (Eigenschaft 1)).

4.2.2 Formulierung mit BERNSTEIN–Polynomen

Statt der rekursiven Beschreibung kann auch eine analytische Beschreibung der BÉZIER–Kurven angegeben werden. Diese erlaubt es in einfacher Weise andere Eigenschaften (wie Stetigkeit, Differenzierbarkeit usw.) zu untersuchen.

Die BERNSTEIN–Polynome vom Grad n sind definiert als

$$B_i^n(t) = \binom{n}{i} t^i (1-t)^{n-i}$$

wobei die Binomialkoeffizienten definiert sind als

$$\binom{n}{i} = \begin{cases} \frac{n!}{i!(n-i)!} & \text{für } 0 \leq i \leq n \\ 0 & \text{sonst} \end{cases}$$

und t ein Parameter im Intervall $[0, 1]$ ist.

Es gelten die folgenden Eigenschaften:

1. Randwert
$$B_0^0(t) = 1$$
2. Beschränkung
$$B_i^n(t) = 0 \text{ für } i \notin \{0, \ldots, n\}$$
3. rekursive Berechnung
$$B_i^n(t) = (1-t)B_i^{n-1}(t) + tB_{i-1}^{n-1}(t)$$
4. Gewichtsfunktion
$$\sum_{i=0}^{n} B_i^n(t) = 1$$

Ein DE CASTELJAU–Punkt, der beim rekursiven Algorithmus berechnet wird, kann mithilfe der BERNSTEIN–Polynome wie folgt angegeben werden:

$$b_i^r(t) = \sum_{j=0}^{r} b_{i+j} B_j^r(t) \qquad 0 \leq t \leq 1$$

was sich durch Induktion beweisen läßt:

$$
\begin{aligned}
b_i^0(t) &= \sum_{j=0}^{0} b_{i+j} B_j^0(t) = b_i B_i^0(t) = b_i \\
b_i^r(t) &= (1-t) b_i^{r-1}(t) + t b_{i+1}^{r-1}(t) \\
&= (1-t) \sum_{j=0}^{r-1} b_{i+j} B_j^{r-1}(t) + t \sum_{j=0}^{r-1} b_{i+1+j} B_j^{r-1}(t) \\
&= (1-t) \sum_{j=i}^{i+r-1} b_j B_{j-i}^{r-1}(t) + t \sum_{j=i+1}^{i+r} b_j B_{j-i-1}^{r-1}(t) \\
&= (1-t) \sum_{j=i}^{i+r-1} b_j B_{j-i}^{r-1}(t) + (1-t) B_r^{r-1}(t) + t \sum_{j=i+1}^{i+r} b_j B_{j-i-1}^{r-1}(t) + t B_{-1}^{r-1}(t) \\
&= (1-t) \sum_{j=i}^{i+r} b_j B_{j-i}^{r-1}(t) + t \sum_{j=i}^{i+r} b_j B_{j-i-1}^{r-1}(t) \\
&= \sum_{j=i}^{i+r} b_j [(t-1) B_{j-i}^{r-1}(t) + t B_{j-i-1}^{r-1}(t)] \\
&= \sum_{j=i}^{i+r} b_j B_{j-i}^{r}(t) \\
&= \sum_{j=0}^{r} b_{i+j} B_{j-i}^{r}(t)
\end{aligned}
$$

Für $r = n$ und $i = 0$ berechnen sich die Punkte der BÉZIER–Kurve als

$$b^n(t) = \sum_{j=0}^{n} b_j B_j^n(t)$$

(Eigenschaft 6)). Wir bezeichnen also mit $b^n(t)$ die BÉZIER–Kurve vom Grad n, die durch den Paramter t über dem Intervall $[0, 1]$ erzeugt wird.

Übungen

1. Beweisen Sie die Eigenschaften der BERNSTEIN–Polynome.
2. Überlegen Sie sich einen Algorithmus, der im zweidimensionalen die Schnittpunkte einer BÉZIER–Kurve, die keine Krümmungswechsel hat, mit einem Strahl berechnet. (Hinweis: es genügt ein Verfahren, das beliebig genau annähern kann.)

Nun wollen wir die Affine Invarianz zeigen (Eigenschaft 8)). Sei A eine $n \times n$–Matrix und v ein n–dimensionaler Vektor. n sei die Dimension des betrachteten Raumes (also für uns entweder 2 oder 3). Sei nun Φ eine affine Transformation, die definiert ist als

$$\Phi(x) = A \cdot x + v$$

Sie transformiert einen Punkt x in einen neuen Punkt $\Phi(x)$. Wendet man nun Φ auf einen Punkt der BÉZIER–Kurve an, erhält man:

$$\Phi(\sum_{j=0}^{n} b_j B_j^n(t)) = A \cdot (\sum_{j=0}^{n} b_j B_j^n(t)) + v$$

$$
\begin{aligned}
&= \sum_{j=0}^{n} B_j^n(t) A \cdot b_j + \sum_{j=0}^{n} B_j^n(t) v \\
&= \sum_{j=0}^{n} B_j^n(t)(A \cdot b_j + v) \\
&= \sum_{j=0}^{n} B_j^n(t) \Phi(b_j)
\end{aligned}
$$

Durch die Verwendung der BERNSTEIN–Polynome erkennt man auch, das die beiden Endpunkte zur Kurve gehören (Eigenschaft 1)), sowie Tatsache, daß für alle anderen Punkte alle Kontrollpunkte für die Berechnung notwendig sind (Eigenschaft 7)).

Die Ableitung des BERNSTEIN–Polynoms B_i^n ist:

$$
\frac{d}{dt} B_i^n(t) = n(B_{i-1}^{n-1}(t) - B_i^{n-1}(t))
$$

Damit gilt für die Ableitung der BÉZIER–Kurve:

$$
\begin{aligned}
\frac{d}{dt} b^n(t) &= n \sum_{j=0}^{n} b_j (B_{j-1}^{n-1}(t) - B_j^{n-1}(t)) \\
&= n \sum_{j=1}^{n} b_j B_{j-1}^{n-1}(t) - n \sum_{j=0}^{n-1} b_j B_j^{n-1}(t) \\
&= n \sum_{j=0}^{n-1} b_{j+1} B_j^{n-1}(t) - n \sum_{j=0}^{n-1} b_j B_j^{n-1}(t) \\
&= n \sum_{j=0}^{n-1} (b_{j+1} - b_j) B_j^{n-1}(t)
\end{aligned}
$$

und man erhält als Ableitung in den Randpunkten durch Einsetzen von $t = 0$ und $t = 1$:

$$
\begin{aligned}
\frac{d}{dt} b^n(0) &= n(b_1 - b_0) \\
\frac{d}{dt} b^n(1) &= n(b_n - b_{n-1})
\end{aligned}
$$

dies gilt, da

$$
B_0^n(0) = B_n^n(t) = 1
$$

was unabhängig von n ist, und da die $B_j^n(t)$ eine Gewichtsfunktion darstellen. Ist somit ein Summand der Summe gleich 1, dann sind alle anderen gleich 0. Dies zeigt, daß die Kurve in den Endpunkten die gleiche Steigung wie die Verbindungsstrecke vom Endpunkt zum Nachbarpunkt hat.

Als Nachteile der BÉZIER–Kurven kann man festhalten:

- Es werden alle Kontrollpunkte bei der Berechnung der Kurvenpunkte benötigt. Dies führt zu Ausgleichseffekten, so daß lokale Änderungen eines Kontrollpunktes nicht die entsprechende, gewünschte Auswirkung auf den Kurvenabschnitt in der Nähe des Kontrollpunktes erzielen.
- Bilden die Kontrollpunkte kein einfaches Polygon kann es zu nicht differenzierbaren Punkten (Spitzen) im Kurvenverlauf kommen.

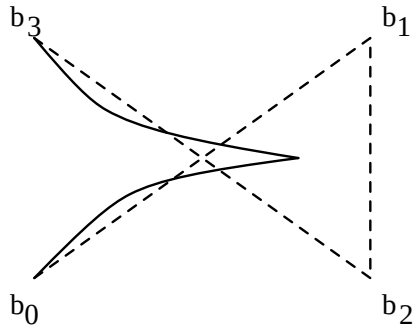

Abbildung: Spitzen in BÉZIER–Kurven

- Möchte man eine vorgegebene Kurve durch eine BÉZIER–Kurve annähern, sind u.U. sehr viele Kontrollpunkte notwendig und deren Auswahl ist nicht einfach.
- Mit steigender Anzahl von Kontrollpunkten steigt auch der Rechenaufwand, der zum Konstruieren der Kurve erforderlich ist.

4.2.3 Abschnittsweise definierte BÉZIER–Kurven

Die Nachteile können durch zusammengesetzte, abschnittsweise berechnete BÉZIER–Kurven überwunden werden. Hierzu verfährt man wie folgt.

Die Gesamtmenge der Kontrollpunkte wird in Gruppen aufgeteilt. Jede Gruppe enthält gleich viele Punkte (meist drei bis fünf) und die ursprüngliche Reihenfolge bleibt bestehen. Jede Gruppe definiert ein Segment der Kurve.

Seien $g_1, \ldots, g_m$ m Gruppen. Jede Gruppe g_i bestehe aus $n+1$ Kontrollpunkten $b_{0i}, \ldots, b_{ni}$. Die Gesamtkurve sei über dem Parameterintervall $[0,1]$ mit Parameter u definiert. Seien die Parameter $u_0, \ldots, u_{m+1}$ eine Zerlegung des Intervalls, wobei gelte:

$$0 = u_0 < u_1 < \ldots < u_m < u_{m+1} = 1$$

Für jedes Intervall $[u_i, u_{i+1}]$ mit $1 \leq i \leq m$ definieren wir einen lokalen Parameter t, der zwischen 0 und 1 läuft, während u zwischen u_i und u_{i+1} läuft:

$$t = \frac{u - u_i}{u_{i+1} - u_i}$$

Das i–te Segment berechnet sich also mithilfe des lokalen Parameters t wie folgt:

$$b_i^n(t) = \sum_{j=0}^{n} b_{ji} B_j^n(t) \qquad \text{mit } t \in [0,1]$$

Um Stetigkeit zu erhalten, d.h.

$$b_i^n(1) = b_{i+1}^n(0) \qquad \text{für } 1 \leq i < m$$

wählt man natürlich

$$b_{ni} = b_{0(i+1)} \qquad \text{für } 1 \leq i < m$$

Um Differenzierbarkeit zu erhalten muß gelten:

$$\left(\frac{d}{du} b_i^n(t)\right)(1) = \left(\frac{d}{du} b_{i+1}^n(t)\right)(0)$$

Mit der Kettenregel erhält man:

$$\begin{aligned}\frac{d}{du}b_i^n(t) &= \frac{d}{dt}b_i^n(t)\frac{dt}{du} \\ &= n\sum_{j=0}^{n-1}(b_{j+1,i}-b_{ji})B_j^{n-1}(t)\frac{1}{u_{i+1}-u_i} \\ \frac{d}{du}b_{i+1}^n(t) &= n\sum_{j=0}^{n-1}(b_{j+1,i+1}-b_{j,i+1})B_j^{n-1}(t)\frac{1}{u_{i+2}-u_{i+1}}\end{aligned}$$

Analog zur obigen Begründung bei der Berechnung der Tangentensteigung in den Endpunkten gilt nun, daß die Ableitungen an den Segmentgrenzen die Bedingung

$$\frac{n}{u_{i+1}-u_i}(b_{ni}-b_{n-1,i}) = \frac{n}{u_{i+2}-u_{i+1}}(b_{1,i+1}-b_{0,i+1})$$

erfüllen müssen. Nun ist $b_{ni} = b_{0(i+1)}$ und man erhält die Bedingung

$$(u_{i+2}-u_{i+1})(b_{ni}-b_{n-1,i}) = (u_{i+1}-u_i)(b_{1,i+1}-b_{ni})$$

Dies bedeutet, daß die drei Punkte $b_{n-1,i}$, $b_{ni} = b_{0,i+1}$ und $b_{1,i+1}$ kollinear sein müssen *und* daß b_{ni} die Strecke $(b_{1,i+1}-b_{n-1,i})$ im Verhältnis $(u_{i+1}-u_i) : (u_{i+2}-u_{i+1})$ teilt. Diese Bedingungen kann man als Nachteil der zusammengefaßten BÉZIER–Kurven ansehen, was bei den B–Spline–Kurven nicht auftritt.

4.3 B–Spline–Kurven

B–Spline–Kurven sind eine Verallgemeinerung der BÉZIER–Kurven oder besser die BÉZIER–Kurven sind ein Speziellfall der B–Spline–Kurven. Die Kurve wird nicht mehr nur durch einen Parameter t aus einem Intervall und den Kontrollpunkten beschrieben, sondern jeder Kontrollpunkt erhält einen weiteren Parameter. Zudem hängen die konstruierten Punkte nicht mehr von allen Kontrollpunkten ab, sondern nur von denen, die sich in einer gewissen Umgebung befinden.

Wir werden die Eigenschaften der B–Spline–Kurven am Ende zusammenfassen.

4.3.1 Der DE BOOR–Algorithmus

Seien $b_0, \ldots, b_n$ eine Menge von Kontrollpunkten und seien $u_0, \ldots, u_{m+1}$ eine Menge von zugehörigen Parameterwerten, für die gelte:

$$0 = u_0 \leq u_1 \leq \cdots \leq u_n = 1$$

Sei u mit $u_0 \leq u \leq u_n$ ein Parameterwert, für den der Punkt der Kurve berechnet werden soll. Damit liegt u in einem Intervall, der durch die u_i vorgegebenen Intervalle, also sei x der entsprechende Index, so daß gelte:

$$u \in [u_x, u_{x+1}[$$

Der DE BOOR–Algorithmus führt nun für eine B–Spline–Kurve vom Grad k $k-1$ Iterationen durch, wobei in Iteration r, d.h. 0¡r¡k, jeweils $k-r$ viele, neue Punkte berechnet werden. Die Rekursionsgleichung lautet mit

$$0 < r < k \qquad \text{und} \qquad x-k+1 \leq i \leq x$$

wie folgt:

$$b_i^r = \frac{u_{i+k-r} - u}{u_{i+k-r} - u_i} b_{i-1}^{r-1} + \frac{u - u_i}{u_{i+k-r} - u_i} b_i^{r-1}$$

wobei wieder die $b_0, \ldots, b_n$ mit den $b_0^0, \ldots, b_n^0$ identifiziert sind. Weiterhin gilt:

$$b_{-1} = b_{-2} = \ldots = b_{-k+1} = 0$$

$$u_{-1} = u_{-2} = \ldots = u_{-k+1} = 0$$

Der Punkt b_x^{k-1} ist der gesuchte Punkt der Kurve.

Daß nun eine BÉZIER–Kurve nur ein Spezialfall einer B–Spline–Kurve ist, sieht man wie folgt ein. Wir setzen

$$\alpha_i^r = \frac{u - u_i}{u_{i+k-r} - u_i}$$

und erhalten damit die Rekursionsgleichung in der Form

$$b_i^r = (1 - \alpha_i^r) b_{i-1}^{r-1} + \alpha_i^r b_i^{r-1}$$

was der Gleichung des DE CASTELJAU–Algorithmus schon sehr ähnelt. Wählen wir weiter

$$k = n + 1$$

$$0 = u_0 < u_1 = \cdots = u_{n-1} = u_n = 1$$

liegt u stets im Intervall $[u_0, u_n]$, d.h. $x = 0$. Somit ist

$$\begin{aligned} u_{i+k-r} &= 1 \\ u_i &= 0 \end{aligned}$$

für alle erlaubten r und i. Damit gilt aber

$$\alpha_i^r = u$$

und man hat den DE CASTELJAU–Algorithmus.

4.3.2 B–Spline–Funktionen

Wir nehmen wieder an, daß

$$0 = u_0 < u_1 < \cdots < u_m = 1$$

eine Zerlegung des Intervalls $[0, 1]$ beschreibt. Ausgehend von der B–Spline–Funktion erster Ordnung

$$N_i^1(u) = \begin{cases} 1 & \text{für } u_i \leq u < u_{i+1} \\ 0 & \text{sonst} \end{cases} \qquad \text{für } 0 \leq i < m$$

definieren wir die *(normalisierten) B–Spline–Funktionen* der Ordnung k durch

$$N_i^k(u) = \frac{u - u_i}{u_{i+k-1} - u_i} N_i^{k-1}(u) + \frac{u_{i+k} - u}{u_{i+k} - u_{i+1}} N_{i+1}^{k-1}(u)$$

für $0 \leq i \leq m - k$.

$N_i^k(u)$ ist auf dem Intervall $[0, 1]$ definiert und beschreibt ein Polynom vom Grad $k - 1$, das nur auf dem offenen Intervall $]u_i, u_{i+k}[$ einen von 0 verschiedenen Wert annimmt, d.h. es gilt:

$$\begin{aligned} N_i^k(u) &= 0 \qquad \text{für } u \leq u_i \quad \text{oder} \quad u \geq u_{i+k} \\ N_i^k(u) &> 0 \qquad \text{für } u_i < u < u_{i+k} \end{aligned}$$

Die Ableitung der B–Spline–Funktionen ergibt sich allgemein zu

$$\frac{d}{du}N_i^k(u) = \frac{k-1}{u_{i+k-1}-u_i}N_i^{k-1}(u) - \frac{k-1}{u_{i+k}-u_{i+1}}N_{i+1}^{k-1}(u) \tag{1}$$

Eine wichtige Eigenschaft der B–Spline–Funktionen ist ihre *lineare Unabhängigkeit,* d.h. wenn

$$\sum_{i=0}^{m-k} c_i N_i^k(u) \equiv 0$$

dann ist $c_i = 0$ für alle i mit $0 \leq i \leq m-k$.

Übungen

1. Beweisen Sie die Eigenschaft der B–Spline–Funktionen, daß sie nur im Intervall $]u_i, u_{i+k}[$ einen von 0 verschiedenen Wert annehmen.
2. Beweisen Sie die lineare Unabhängigkeit der B–Spline–Funktionen.

Die normalisierten B–Spline–Funktionen haben auch die für Gewichtsfunktionen typische Eigenschaft

$$\sum_{i=0}^{m-k} N_i^k(u) = 1 \qquad \text{für } u \in [u_{k-1}, u_{m-k+1}]$$

Einen Spezialfall der normalisierten B–Spline–Funktionen erhält man, wenn man die Intervallpunkte $u_0, \cdots, u_m$ äquidistant zwischen 0 und 1 verteilt, d.h. wenn man

$$u_i = i/m$$

setzt. Die Definitionsgleichung wird dann zu:

$$N_i^k(u) = \frac{mu-i}{k-1}N_i^{k-1}(u) + \frac{i+k-mu}{k-1}N_{i+1}^{k-1}(u)$$

Die so entstehenden B–Spline–Funktionen nennt man *uniforme* B–Spline–Funktionen. Setzen wir $x = mu$ erhalten wir:

$$N_i^k(x) = \frac{x-i}{k-1}N_i^{k-1}(x) + \frac{i+k-x}{k-1}N_{i+1}^{k-1}(x)$$

Bemerkung: Eigentlich müßte man $N_i^k(x/m)$ schreiben. Dies ist jedoch wegen

$$\begin{aligned} N_i^1(\frac{x}{m}) &= \begin{cases} 1 & \text{für } \frac{i}{m} = u_i \leq \frac{x}{m} < u_{i+1} = \frac{i+1}{m} \\ 0 & \text{sonst} \end{cases} \\ &= \begin{cases} 1 & \text{für } i \leq x < i+1 \\ 0 & \text{sonst} \end{cases} \end{aligned}$$

unabhängig von m und demzufolge kann man statt x/m einfach x schreiben.

Damit erhält man für N_0^2:

$$N_0^2(x) = xN_0^1(x) + (2-x)N_1^1(x) = \begin{cases} x & \text{für } x \in [0,1[\\ 2-x & \text{für } x \in [1,2[\end{cases}$$

und allgemein für N_i^2:

$$N_i^2(x) = \begin{cases} x - i & \text{für } x \in [i, i+1[\\ 2 + i - x & \text{für } x \in [i+1, i+2[\end{cases}$$

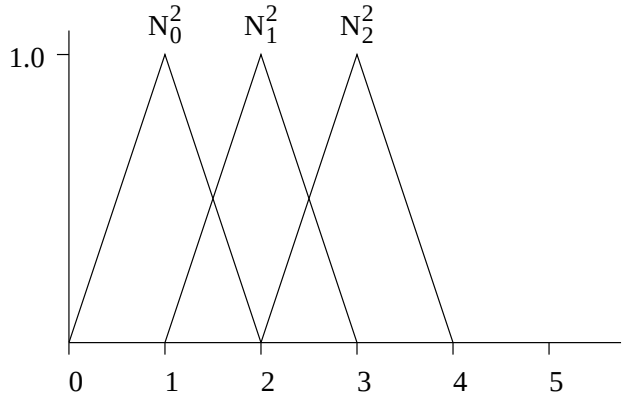

Abbildung: Darstellung von $N_i^2(x)$

Wie man in der Abbildung sieht, entsteht N_{i+1}^2 aus N_i^2 durch eine Verschiebung um 1 nach rechts, d.h. durch eine Translation des Argumentwertes um $\Delta x = -1$. Es ist also:

$$N_{i+1}^2(x) = N_i^2(x-1) = \cdots = N_0^2(x-i-1)$$

Diese Translationsinvarianz kann man durch die folgende Parametertransformation erfassen:

$$w = x \pmod 1$$

d.h. für $x \in [0,1[$ ist $x = w$
für $x \in [1,2[$ ist $x = w + 1$
für $x \in [2,3[$ ist $x = w + 2$
...

wobei jeweils $w \in [0,1[$. Damit erhält man:

$$\begin{aligned} N_0^2(w) &= wN_0^1(w) + (1-w)N_1^1(w) \\ N_1^2(w) &= wN_1^1(w) + (1-w)N_2^1(w) \end{aligned}$$

und allgemein

$$N_i^2(w) = wN_i^1(w) + (1-w)N_{i+1}^1(w)$$

Für $k = 3$ erhält man:

$$\begin{aligned} N_0^3(x) &= \frac{x}{2}N_0^2(x) + \frac{3-x}{2}N_1^2(x) \\ &= \frac{x}{2}(wN_0^1(w) + (1-w)N_1^1(w)) + \frac{3-x}{2}(wN_1^1(w) + (1-w)N_2^1(w)) \end{aligned}$$

Nach der Parametertransformation wird daraus:

$$\begin{aligned} N_0^3(w) &= \frac{w^2}{2}N_0^1(w) + \frac{(1+w)(1-w) + (2-w)w}{2}N_1^1(w) + \frac{(1-w)^2}{2}N_2^1(w) \\ &= \frac{w^2}{2}N_0^1(w) + \frac{-2w^2 + 2w + 1}{2}N_1^1(w) + \frac{(1-w)^2}{2}N_2^1(w) \end{aligned}$$

und allgemein

$$N_i^3(w) = \frac{w^2}{2}N_i^1(w) + \frac{-2w^2 + 2w + 1}{2}N_{i+1}^1(w) + \frac{(1-w)^2}{2}N_{i+2}^1(w)$$

was in folgender Abbildung veranschaulicht ist:

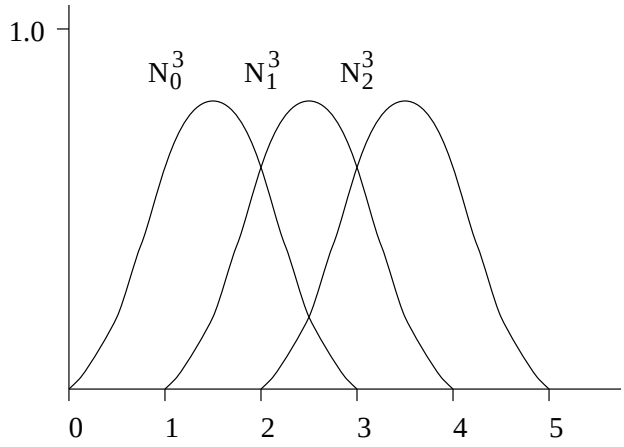

Abbildung: Darstellung von $N_i^3(x)$

Analog erhält man:

$$\begin{aligned} N_i^4(w) &= \frac{w^3}{6}N_i^1(w) + \frac{3w^3+3w^2+3w+1}{6}N_{i+1}^1(w) + \\ &\quad \frac{3w^3-6w^2+4}{6}N_{i+2}^1(w) + \frac{(1-w)^3}{6}N_{i+3}^1(w) \end{aligned}$$

4.3.3 Formulierung mit B–Spline–Funktionen

Analog zu den BÉZIER–Kurven, die mit BERNSTEIN–Polynomen definiert worden sind, werden die B–Spline–Kurven mithilfe der B–Spline–Funktionen definiert.

Seien $b_0, \ldots, b_n$ die Kontrollpunkte mit $n > k$. Die B–Spline–Kurve vom Grad k ist definiert durch:

$$s_k(u) = \sum_{i=0}^{n} b_i N_i^k(u + \frac{k}{2(n+k)}) \qquad \text{für} \qquad u_0 \leq u \leq u_n$$

dabei nehmen wir uniforme B–Spline–Funktionen an, d.h. es gilt

$$0 = u_0 < \ldots < u_i = \frac{i}{n+k} < \ldots < u_{n+k} = 1$$

Es werden mehr als n Parameterwerte benötigt (Definition der B–Spline–Funktionen).

Die Addition von $\frac{k}{2(n+k)}$ zum Parameterwert u dient dazu, die verwendete B–Spline–Funktion an die richtige Stelle zu rücken: Die Funktion $N_i^k(u)$ nimmt ihr Maximum für $u = u_i + \frac{k}{2(n+k)}$ an. Die Addition von $\frac{k}{2(n+k)}$ bewirkt ein Verschieben der Funktion um $\frac{k}{2(n+k)}$ nach links, das Maximum wird für $u = u_i$ angenommen. Das bewirkt, daß der Kontrollpunkt b_i zur Berechnung der B–Spline–Kurve an dem b_i zugeordneten Intervallpunkt u_i maximal berücksichtigt wird.

Übungen

1. Formulieren Sie die Berechnung einer kubischen B–Spline–Kurve ($k = 4$) unter Auflösung der Summennotation und Berücksichtigung der Parametertransformationen, d.h. geben sie die (einfache) Berechnungsformel der Kurvenpunkte an.

Die Ableitung der B–Spline–Kurven ergibt sich zu

$$\frac{d}{du}s_k(u) = (k-1)\sum_{i=1} n\frac{b_i - b_{i-1}}{u_{i+k-1} - u_i}N_i^{k-1}(u)$$

4.3.4 Interpolation der Randpunkte einer B–Spline–Kurve

B–Spline–Kurven interpolieren nicht die Randpunkte, d.h. b_0 und b_n liegen nicht auf der durch obiges Polynom beschriebenen Kurve. Dies ist beim Modellieren von Nachteil, da man dies oft als Unterstützung bei der Konstruktion von Kurven wünscht.

Man erreicht eine Interpolation der Randpunkte durch zwei Vorgehensweisen: Vervielfältigung der End–Kontrollpunkte (bis zu $(k-1)$ mal), oder Einführung von Phantompunkten.

Mehrfachpunkte Durch eine $(k-1)$–fache Vervielfältigung der End–Kontrollpunkte erreicht man, daß die entstehenden B–Spline–Kurven auf jeden Fall durch die End–Kontrollpunkte verlaufen. Man führt Vervielfältigung neue Kontrollpunkte

$$b_{-k+1} = b_{-k+2} = \cdots = b_{-1} = b_0$$
$$b_{n+1} = b_{n+2} = \cdots = b_{n+k-1} = b_n$$

ein und legt die geänderte Parameterzerlegung

$$0 = u_{-(k-1)} < \ldots < u_i = \frac{i+k-1}{m} < \ldots < u_{n+k+k-1} = 1$$

zugrunde mit,

$$m = n + k + 2(k-1) = n + 3k - 1$$

Die B–Spline–Kurve berechnet sich dann als:

$$s_k(u) = \sum_{i=-k+1}^{n+k-1} b_i N_i^k(u + \frac{k}{2m}) \qquad \text{für } u_0 \le u \le u_n$$

Daß diese Kurve die End–Kontrollpunkte interpoliert, folgt aus der Eigenschaft der B–Spline–Kurven, daß zur Berechnung des Wertes an einer bestimmten Parameterposition höchstens $k-1$ Kontrollpunkte berücksichtigt werden. Man beachte, daß wir den gleichen Definitionsbereich für u verwenden, obwohl wir jetzt zusätzliche Intervallpunkte eingefügt haben. Die zusätzlichen Kontrollpunkte dienen nur dazu, die B–Spline–Kurve in der Nähe der ursprünglichen End–Kontrollpunkte das gewünschte Aussehen zu geben.

Das Verfahren der Vervielfältigung der End–Kontrollpunkte hat den Nachteil, daß dadurch in der entstehenden B–Spline–Kurve Unstetigkeiten auftreten können. Dies läßt sich mindern, wenn man die End–Kontrollpunkte weniger als $k-1$ Mal vervielfältigt. Dann interpoliert die B–Spline–Kurve zwar die End–Kontrollpunkte nicht immer, aber die B–Spline–Kurve nähert sich den End–Kontrollpunkte doch sehr stark.

Phantompunkte Wir betrachten eine kubische B–Spline–Kurve. Gesucht ist ein (Phantom–)Punkt b_{-1} mit der Eigenschaft:

$$s_k(u_0) = \frac{1}{6}b_{-1} + \frac{2}{3}b_0 + \frac{1}{6}b_1 = b_0$$

Dies erreicht man mit

$$b_{-1} = 2b_0 - b_1$$

und analog für den anderen Endpunkt

$$b_{n+1} = 2b_n - b_{n-1}$$

Die Berechnungsvorschrift lautet nun

$$s_k(u) = \sum_{i=-1}^{n+1} b_i N_{i+k-1}^k(u + \frac{k}{2m}) \qquad \text{für } u_0 \le u \le u_n$$

wobei die Parameterzerlegung

$$0 = u_{-1)} < \ldots < u_i = \frac{i+1}{m} < \ldots < u_{n+k+1} = 1$$

zugrunde liegt mit

$$m = n + k + 2$$

Der Definitionsbereich bleibt wiederum unverändert.

4.3.5 Geschlossene B–Spline–Kurven

Durch Gleichsetzen der beiden End–Kontrollpunkte erreicht man, daß die Kurve geschlosen ist. Allerdings entsteht einerseits eine Spitze (d.h. die Kurve ist nicht differenzierbar) und andererseits ist die Kurve nicht notwendigerweise symmetrisch, auch wenn dies die Lage der Kontrollpunkte erwarten ließe. Allgemein läßt sich dieser Defekt beheben, indem man die Kontrollpunkte in beide Richtungen periodisch fortsetzt. Wenn man eine geschlossene B–Spline–Kurve erhalten will, setzt man also:

$$b_{-1} = b_n, b_{-2} = b_{n-1}, \cdots, b_{-(k-1)} = b_{n-k}$$
$$b_{n+1} = b_1, b_{n+2} = b_2, \cdots, b_{n+k-1} = b_{k-1}$$

Dabei verwendet man die gleiche Parameterzerlegung wie bei der $(k-1)$–fachen Vervielfältigung der Kontrollpunkte.

4.3.6 Zusammenfassung der Eigenschaften

Am Ende wollen wir die Eigenschaften der B–Spline–Kurven nochmals zusammenfassen, wobei wir jedoch nicht alle diese Eigenschaften oben gezeigt haben:

1. Sie verläuft im Normalfall *nicht* durch die Endpunkte b_0 und b_n.
2. Dies kann jedoch durch Mehrfachpunkte oder Phantompunkte korrigiert werden.
3. Sie liegt vollständig in der konvexen Hülle der Kontrollpunkte.
4. Es gilt sogar die stärkere Eigenschaft, daß das zu dem Intervall $I = [u_x, u_{x+k-1}](0 \leq x \leq n-(k-1))$ gehörende Kurvensegment ganz in der konvexen Hülle der zugehörigen Kontrollpunkte $b_x, \cdots, b_{x+k-1}$ liegt.
5. Sie können durch einen rekursiven Algorithmus (den DE BOOR–Algorithmus) oder
6. durch Funktionen (die B–Spline–Funktionen) beschrieben werden.
7. Nur k–Kontrollpunkte sind notwendig, um einen Punkt auf der BÉZIER–Kurve zu berechnen.
8. Demzufolge können die Kurven lokal modelliert werden.
9. Es sind sehr leicht auch geschlossene Kurven zu modellieren.
10. Sie sind invariant gegenüber affinen Transformationen.
11. Sie sind nicht invariant gegenüber Projektionen.
12. Eine B–Spline–Kurve verläuft üblicherweise dicht in der Nähe des Mittelpunkts jeder Kante ihres Kontrollpunkt–Polygons.

13. Wenn $k-1$ benachbarte Kontrollpunkte kollinear liegen, so berührt die B–Spline–Kurve das Kontrollpunkt–Polygon zwischen den kollinearen Kontrollpunkten.

14. Wenn k benachbarte Kontrollpunkte kollinear liegen, so ist das zugehörige B–Spline–Kurvensegment eine Gerade.

15. Wenn $k-1$ Kontrollpunkte zusammenfallen, so kann die B–Spline–Kurve eine Spitze besitzen.

4.4 Interpolierende Kurven

In vielen Situationen ist es erforderlich, daß die modellierte Kurve wirklich durch die Kontrollpunkte hindurch verläuft. Wir betrachten wieder eine Menge von Kontrollpunkten $b_0, \ldots, b_n$ die wir durch die Parameter t_i äquidistant im Intervall $[0,1]$ anordnen, d.h. der Kontrollpunkt b_i ist dem Parameterwert $t_i = i/n$ zugeordnet. Gesucht ist eine Kurve p, die die Kontrollpunkte interpoliert, d.h. für die gilt:

$$p(t_i) = b_i \qquad \text{für} \qquad 1 \leq i \leq n$$

4.4.1 LAGRANGE–Interpolation

Wir werden hier nur das Verfahren der LAGRANGEschen Interpolation kurz vorstellen, das die gegebenen $n+1$ Punkte durch ein Polynom n–ten Grades verbindet, um die Probleme der polynomiellen Interpolation aufzuzeigen. Für weitergehende Verfahren sei auf die Literatur verwiesen.

Für zwei Kontrollpunkte b_i und b_{i+1} mit den zugehörigen Parameterwerten t_i und t_{i+1}, beschreibt

$$p_i^1(t) = \frac{t_{i+1}-t}{t_{i+1}-t_i} b_i + \frac{t-t_i}{t_{i+1}-t_i} b_{i+1} \qquad 1 \leq i \leq n-1$$

die durch b_i und b_{i+1} verlaufende Gerade, d.h. das Polynom vom Grad 1, das b_i und b_{i+1} interpoliert.

Aufbauend auf dieser Formel können wir rekursiv das Polynom vom Grad i definieren, das $i+1$ Kontrollpunkte interpoliert. Sei p_0^{i-1} das Polynom vom Grad $i-1$, das $b_0, \cdots, b_{i-1}$ interpoliert und sei p_1^{i-1} das Polynom vom Grad $i-1$, das $b_1, \cdots, b_i$ interpoliert, dann ist

$$p_0^i(t) = \frac{t_i - t}{t_i - t_0} p_0^{i-1}(t) + \frac{t-t_0}{t_i-t_0} p_1^{i-1}(t)$$

das Polynom, das $b_0, \cdots, b_i$ interpoliert. Daß dies gilt, sieht man durch Einsetzen der Werte $t_0, \cdots, t_i$:

$$\begin{aligned} p_0^i(t_0) &= 1 * p_0^{i-1}(t_0) + 0 * p_1^{i-1}(t_0) = p_0 \\ p_0^i(t_i) &= 0 * p_0^{i-1}(t_0) + 1 * p_1^{i-1}(t_0) = p_i \end{aligned}$$

Außerdem gilt für alle Parameterwerte t_j

$$p_0^{i-1}(t_j) = p_1^{i-1}(t_j) = p_j \quad \text{mit} \quad 1 \leq j < i$$

Daraus folgt dann durch Einsetzen auch $p_0^i(t_j) = p_j$. Damit können wir das die $n+1$ Kontrollpunkte interpolierende Polynom wie folgt rekursiv definieren: ausgehend von $p_i^0 = b_i$ setzen wir

$$p_i^r(t) = \frac{t_{i+r}-t}{t_{i+r}-t_i} p_i^{r-1}(t) + \frac{t-t_i}{t_{i+r}-t_i} p_{i+1}^{r-1}(t)$$

für $1 \leq r \leq n$, $0 \leq i \leq n-r$. Diese rekursive Formulierung ist auch als AITKEN's Algorithmus bekannt. Das gesuchte Polynom ist p_0^n.

Eine alternative Formulierung des interpolierenden Polynoms kann mit Hilfe der LAGRANGEschen Polynome erfolgen. Dies hat den Vorteil, daß die Kontrollpunkte explizit enthalten sind.

Die LAGRANGEschen Polynome L_i^n sind definiert als

$$L_i^n(t) = \prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{(t - t_j)}{(t_i - t_j)}$$

Und man erhält das die Kontrollpunkte interpolierenden Polynom als

$$p(t) = \sum_{i=0}^{n} b_i L_i^n(t)$$

Es gilt

$$L_i^n(t_j) = \delta_{i,j}$$

wobei $\delta_{i,j}$ das sogenannte KRONECKER–Symbol ist, mit

$$\delta_{i,j} = \begin{cases} 1 & \text{für } i = j \\ 0 & \text{sonst} \end{cases}$$

d.h. für den Parameterwert t_i verschwinden alle LAGRANGEschen Polynome außer L_i^n und es ist $L_i^n(t_i) = 1$, woraus folgt, daß die Kontrollpunkte interpoliert werden.

Die Eindeutigkeit des interpolierenden Polynoms folgt aus der Tatsache, daß es nur ein Polynom vom Grad n gibt, das $n + 1$ Kontrollpunkte interpoliert. Obige Formel ist die eindeutige Darstellung dieses Polynoms mit den LAGRANGEschen Polynomen als Basis. Der vorher beschriebene AITKENS–Algorithmus liefert somit das gleiche Polynom.

Der wesentliche Nachteil der interpolierenden Polynome sind deren Oszillationen, die auftreten können sobald mehrere Punkte interpoliert werden sollen. Sie sind also nur für relativ kleine Punktmengen einzusetzen.

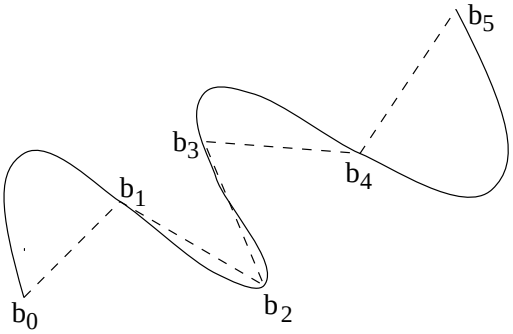

Abbildung: Oszillierende Interpolationspolynome

4.4.2 B–Spline–Interpolation

Wir werden die vorgestellten B–Spline–Kurven nun auch verwenden, um Kontrollpunkte zu interpolieren. Um die B–Spline–Kurven zur Interpolation von $b_0, \ldots, b_n$ verwenden zu können, muß man neue Kontrollpunkte $x_0, \ldots, x_n$ einführen, die so zu wählen sind, daß die $b_0, \ldots, b_n$ interpoliert werden. Dieses Vorgehen ähnelt dem beschriebenen Phantompunkt–Verfahren,

Wir nehmen wieder an, daß die Interpolationspunkte mithilfe eines Parameters äquidistant auf $[0, 1]$ angeordnet sind, d.h. b_i ist der Parameterwert $u_i = i/m$ mit $m = n + k$ zugeordnet. Damit die $b_0, \ldots, b_n$ interpoliert werden, muß gelten:

$$b_j = s_k(u_i) = \sum_{i=0}^{n} x_i N_i^k(u_j + \frac{k}{2m}) \qquad \text{für } 0 \leq j \leq n$$

Dies ist ein System von $n+1$ linearen Gleichungen mit $n+1$ Unbekannten $x_0, \ldots, x_n$, das mit einem gängigen Verfahren (GAUSS'sche Elimination oder GAUSS–SEIDEL–Iteration) gelöst werden kann. In der Praxis verwendet man häufig kubische B–Spline–Kurven ($k = 4$) und man erhält

$$\begin{aligned} b_j &= \sum_{i=0}^{n} x_i N_i^4(u_j + \frac{2}{m}) \\ &= \frac{1}{6}x_{j-1} + \frac{2}{3}x_j + \frac{1}{6}x_{j+1} \end{aligned}$$

wobei $x_{-1} = x_{n+1} = 0$ gesetzt ist. Dies läßt sich in Matrixform als

$$\mathbf{A}x = b$$

darstellen und man erhält ausgeschrieben:

$$\begin{bmatrix} \frac{2}{3} & \frac{1}{6} & 0 & 0 & \cdots & 0 & 0 & 0 \\ \frac{1}{6} & \frac{2}{3} & \frac{1}{6} & 0 & \cdots & 0 & 0 & 0 \\ 0 & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} & \cdots & 0 & 0 & 0 \\ & & \ddots & \ddots & \ddots & & & \\ 0 & 0 & 0 & 0 & \cdots & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \\ 0 & 0 & 0 & 0 & \cdots & 0 & \frac{1}{6} & \frac{2}{3} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ \vdots \\ b_{n-1} \\ b_n \end{bmatrix}$$

Wenn man die inverse Matrix $\mathbf{A}^{-1}$ zu $\mathbf{A}$ kennt, lassen sich die unbekannten Kontrollpunkte auch direkt durch

$$x = \mathbf{A}^{-1}b$$

berechnen. Da $\mathbf{A}$ nicht von den Interpolationspunkten abhängt, kann man $\mathbf{A}^{-1}$ vorberechnen. Mit den neuen Kontrollpunkte $x_0, \ldots, x_n$ läßt sich die interpolierende B–Spline–Kurve wie gehabt bestimmen:

$$s_k(u) = \sum_{i=0}^{n} x_i N_i^k(u + \frac{k}{2m}) \qquad \text{für} \qquad u_0 \le u \le u_n$$

4.4.3 Korrektur des Verlaufs der interpolierenden B–Spline–Kurven an den Endpunkten

Man sieht an diesen Beispielen, daß Manche der interpolierenden B–Spline–Kurven zeigen in der Nähe der End–Interpolationspunkte nicht ganz befriedigende Ergebnisse. Dies läßt sich beheben, indem wir zwei zusätzliche Kontrollpunkte x_{-1} und x_{n+1} aufnehmen, die dazu dienen, die entstehenden Kurven in der Nähe der End–Interpolationspunkte zu kontrollieren. Man erhält:

$$s_k(u) = \sum_{i=-1}^{n+1} x_i N_i^k(u + \frac{k}{2m}) \qquad \text{für} \qquad u_0 \le u \le u_n$$

Eine Möglichkeit der Kontrolle ist es, von den Kurven zu verlangen, daß sie an den End–Interpolationspunkten die gleiche Steigung haben wie der durch die Interpolationspunkte verlaufende Polygonzug.

Seien also $\vec{t}_0 = b_1 - b_0$ und $\vec{t}_n = b_n - b_{n-1}$ die vorgegebenen Tangentenvektoren. Für die Ableitung der kubischen B–Spline–Kurven erhält man durch direktes Differenzieren:

$$\frac{ds_k}{du} = \sum_{i=-1}^{n+1} x_i \frac{dN_i^k}{du}(u + \frac{k}{2m}) = \sum_{i=-1}^{n+1} x_i \frac{dN_i^4}{du}(u + \frac{k}{2m})$$

wobei

$$\begin{aligned}\frac{dN_i^4}{du}(u) &= \frac{dN_i^4}{dw}\frac{dw}{dx}\frac{dx}{du} = m\frac{dN_i^4}{dw} \\ &= m\left(\frac{3w^2}{6}N_i^1(w) + \frac{9w^2+6w+3}{6}N_{i+1}^1(w) + \right. \\ &\qquad \left. \frac{9w^2-12w}{6}N_{i+2}^1(w) + \frac{-3(1-w)^2}{6}N_{i+3}^1(w)\right)\end{aligned}$$

Damit erhält man

$$\begin{aligned}\frac{ds_4}{du}(u_0) &= mx_{-1}\frac{dN_{-1}^4}{dw}(0) + mx_0\frac{dN_0^4}{dw}(0) + mx_1\frac{dN_1^4}{dw}(0) \\ &= -\frac{m}{2}x_{-1} + \frac{m}{2}x_1\end{aligned}$$

und analog

$$\frac{ds_4}{du}(u_n) = -\frac{m}{2}x_{n-1} + \frac{m}{2}x_{n+1}$$

Daraus erhält man dann folgendes Gleichungssystem:

$$\begin{bmatrix} -\frac{m}{2} & 0 & \frac{m}{2} & 0 & \cdots & 0 & 0 & 0 \\ \frac{1}{6} & \frac{2}{3} & \frac{1}{6} & 0 & \cdots & 0 & 0 & 0 \\ 0 & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} & \cdots & 0 & 0 & 0 \\ & & \ddots & \ddots & \ddots & & & \\ 0 & 0 & 0 & 0 & \cdots & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \\ 0 & 0 & 0 & 0 & \cdots & -\frac{m}{2} & 0 & \frac{m}{2} \end{bmatrix} \begin{bmatrix} x_{-1} \\ x_0 \\ x_1 \\ \vdots \\ x_n \\ x_{n+1} \end{bmatrix} = \begin{bmatrix} \vec{t}_0 \\ b_0 \\ b_1 \\ \vdots \\ b_n \\ \vec{t}_n \end{bmatrix}$$

4.5 Oberflächen

Bei den Oberflächen beschränken wir uns auf die approximierenden Verfahren.

4.5.1 BÉZIER–Oberflächen

Eine gekrümmte Oberfläche entsteht durch Verschieben einer Kurve durch den Raum, die sich während dieser Verschiebung ihrerseits modifiziert.

Eine BÉZIER–Oberfläche entsteht, wenn sich die Kontrollpunkte eines BÉZIER–Kurve auf BÉZIER–Kurven bewegen. Hierzu nehmen wir an, daß die sich bewegende Kurve vom Grad m ist, d.h. mit den Kontrollpunkten $b_0, \ldots, b_m$ konstuiert ist, und über dem Parameter u definiert wird:

$$b^m(u) = \sum_{i=0}^{m} b_i B_i^m(u) \qquad \text{für } 0 \le u \le 1$$

Jeder Kontrollpunkt b_i durchläuft eine BÉZIER–Kurve vom gleichen Grad n über einem Parameter v, d.h. es ist

$$b_i = b_i(v) = \sum_{j=0}^{n} b_{ij} B_j^n(v) \qquad \text{für } 0 \le v \le 1$$

wobei $b_{i0}, \ldots, b_{in}$ die Kontrollpunkte für diese BÉZIER–Kurven sind. Ein Einsetzen liefert als Gleichung für einen Punkt auf der BÉZIER–Oberfläche

$$b^{m,n}(u,v) = \sum_{i=0}^{m} \sum_{j=0}^{n} b_{ij} B_i^m(u) B_j^n(v) \qquad \text{für } 0 \leq u, v \leq 1$$

was ein Tensorprodukt darstellt.

Die $(n+1)(m+1)$ Kontrollpunkte b_{ij} spannen ein Netz auf, in dem jeder Kontrollpunkt mit seinen vier Nachbarpunkten verbunden ist (man beachte Rand– und Eckpunkte).

Wenn man den Parameterwert $v = v_{const}$ konstant hält, erhält man einen Schnitt (auch *Parameterlinie* genannt) durch die BÉZIER–Oberfläche, der wieder eine BÉZIER–Kurve ist. Die Kontrollpunkte dieser BÉZIER–Kurve sind

$$b_i(v_{const}) = \sum_{j=0}^{n} b_{ij} B_j^n(v_{const}) \qquad \text{für } 0 \leq i \leq m$$

Da diese Punkte auf BÉZIER–Kurven liegen, lassen sie sich natürlich auch mit Hilfe des DE CASTELJAU–Algorithmus berechnen. Ausgehend von diesen Kontrollpunkten kann man die Punkte auf der Schnittkurve ebenfalls mit Hilfe des des DE CASTELJAU–Algorithmus berechnen, diesmal auf den Parameterwert u angewendet. Analoges gilt für konstantes $u = u_{const}$.

Die meisten Eigenschaften der BÉZIER–Kurven lassen sich auf die BÉZIER–Oberflächen übertragen. Insbesondere gelten folgende Vor– und Nachteile:

1. Die Oberfläche liegt vollständig in der konvexen Hülle der Kontrollpunkte.
2. Die Eckpunkte werden interpoliert.
3. Sie ist gegenüber affinen Transformationen invariant.
4. Zur Berechnung eines Oberflächenpunktes werden alle Kontrollpunkte benötigt.
5. Lokale Änderungen sind deshalb nicht sehr leicht möglich.
6. Vorteilhaft ist jedoch, daß sie deshalb im allgemeinen sehr glatt aussehen.
7. Auch BÉZIER–Oberflächen können abschnittsweise definiert und dann zusammengesetzt werden.

Wir hatten als Ableitung einer BÉZIER–Kurve

$$\frac{d}{dt} b^n(t) = n \sum_{j=0}^{n-1} (b_{j+1} - b_j) B_j^{n-1}(t)$$

und erhalten somit die partiellen Ableitungen einer BÉZIER–Oberfläche einfach als:

$$\begin{aligned} \frac{\partial}{\partial u} b^{m,n}(u,v) &= m \sum_{i=0}^{m-1} \sum_{j=0}^{n} (b_{i+1,j} - b_{ij}) B_i^{m-1}(u) B_j^n(v) \\ \frac{\partial}{\partial v} b^{m,n}(u,v) &= n \sum_{i=0}^{m} \sum_{j=0}^{n-1} (b_{i,j+1} - b_{ij}) B_i^m(u) B_j^{n-1}(v) \end{aligned}$$

woraus sich der Normalenvektor durch das Kreuzprodukt berechnet:

$$\vec{n}(u,v) = \frac{\partial}{\partial u} b^{m,n}(u,v) \times \frac{\partial}{\partial v} b^{m,n}(u,v)$$

Das Tensorprodukt läßt sich auch in Matrixform darstellen

$$b^{m,n}(u,v) = [B_0^m(u) \cdots B_m^m(u)] \begin{bmatrix} b_{00} & \cdots & b_{0n} \\ \vdots & & \vdots \\ b_{m0} & \cdots & b_{mn} \end{bmatrix} \begin{bmatrix} B_0^n(v) \\ \vdots \\ B_n^n(v) \end{bmatrix}$$

4.5.2 B–Spline–Oberflächen

B–Spline–Oberflächen werden analog zu BÉZIER–Oberflächen allerdings nicht mit BERNSTEIN–Polynomen sondern mit B–Spline–Funktionen beschrieben:

$$s_{kl}(u,v) = \sum_{i=0}^{m} \sum_{j=0}^{n} b_{ij} N_i^k(u + \frac{k}{2x}) N_j^l(v + \frac{l}{2y})$$

Dabei setzen wir wieder eine uniforme Parameterzerlegung, jetzt in beiden Richtungen voraus, d.h. es gilt $u_i = i/x$ und $v_j = j/y$ mit $x = m + k$ und $y = n + l$. Nun ist jedem Kontrollpunkt b_{ij} ist zweidimensionaler Parameterpunkt (u_i, v_j) zugeordnet.

Auch hier lassen sich die Eigenschaften der B–Spline–Kurven im wesentlichen übertragen, insbesondere die der lokalen Kontrolle. Zur Berechnung eines Punktes der Oberfläche werden nur $k * l$ Kontrollpunkte herangezogen.

Als Ableitung der B–Spline–Kurven hatten wir

$$\frac{d}{du} s_k(u) = (k-1) \sum_{i=1} n \frac{b_i - b_{i-1}}{u_{i+k-1} - u_i} N_i^{k-1}(u)$$

Für die partiellen Ableitungen der B–Spline–Oberflächen ergibt sich somit analog zu den BÉZIER–Oberflächen:

$$\begin{aligned} \frac{\partial}{\partial u} s_{kl}(u,v) &= (k-1) \sum_{i=1}^{m} \sum_{j=0}^{n} \frac{b_{ij} - b_{i-1,j}}{u_{i+k-1} - u_i} N_i^{k-1}(u + \frac{k}{2x}) N_j^l(v + \frac{l}{2y}) \\ \frac{\partial}{\partial v} s_{kl}(u,v) &= (l-1) \sum_{i=0}^{m} \sum_{j=1}^{n} \frac{b_{ij} - b_{i,j-1}}{v_{j+l-1} - v_i} N_i^k(u + \frac{k}{2x}) N_j^{l-1}(v + \frac{l}{2y}) \end{aligned}$$

Im folgenden werden wir uns auf den in der Praxis wichtigen Fall der bikubischen B–Spline–Oberflächen beschränken ($k = l = 4$). In diesem Fall vereinfacht sich die Formel zu

$$s(u,v) = \sum_{i=0}^{m} \sum_{j=0}^{n} b_{ij} N_i^4(u + \frac{2}{x}) N_j^4(v + \frac{2}{y})$$

für $u_0 \le u \le u_m$ und $v_0 \le v \le v_n$.

In Matrixschreibweise ergibt sich:

$$s(u,v) = \left[N_0^4(u + \frac{2}{x}) \ \cdots \ N_m^4(u + \frac{2}{x}) \right] \begin{bmatrix} b_{00} & \cdots & b_{0n} \\ \vdots & & \vdots \\ b_{m0} & \cdots & b_{mn} \end{bmatrix} \begin{bmatrix} N_0^4(v + \frac{2}{y}) \\ \vdots \\ N_n^4(v + \frac{2}{y}) \end{bmatrix}$$

Ein Punkt im bikubischen Fall berechnet sich — ähnlich wie bei den Kurven — allein aus $16 = k * l$ vielen Kontrollpunkten. Für einen zu einem Kontrollpunkt b_{ab} gehörenden Parameterpunkt (u_a, v_b)

genügen sogar 9 Kontrollpunkte. Analog zu der Rechnung für Kurven ergibt sich

$$\begin{aligned} s(u_a, v_b) &= \begin{bmatrix} \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \end{bmatrix} \begin{bmatrix} b_{a-1,b-1} & b_{a-1,b} & b_{a-1,b+1} \\ b_{a,b-1} & b_{a,b} & b_{a,b+1} \\ b_{a+1,b-1} & b_{a+1,b} & b_{a+1,b+1} \end{bmatrix} \begin{bmatrix} \frac{1}{6} \\ \frac{2}{3} \\ \frac{1}{6} \end{bmatrix} \\ &= \frac{1}{36} \left(b_{a-1,b-1} + b_{a-1,b+1} + b_{a+1,b-1} + b_{a+1,b+1}\right) + \\ &\quad \frac{1}{9} \left(b_{a-1,b} + b_{a,b-1} + b_{a,b+1} + b_{a+1,b}\right) + \\ &\quad \frac{4}{9} \, b_{a,b} \end{aligned}$$

4.5.3 Interpolation der Randpunkte einer B–Spline–Oberfläche

Bei den B–Spline–Oberflächen stellt sich auch das Problem, daß die Eckpunkte nicht interpoliert werden. Mehrnoch, die Flächen zeigen an den Rändern oft nicht den gewünschten glatten Verlauf.

Dies läßt sich wiederum einerseits durch Mehrfachpunkte oder durch Phantompunkte beheben.

Mehrfachpunkte Oft genügt es die Randkontrollpunkte zu verdoppeln, d.h. man fügt $2(n+1+m+1)+4$ zusätzliche Randpunkte ein, die das Kontrollpunkt–Netz vollständig umgeben. Die neuen Randpunkte führen nun dazu, daß die Ränder der Fläche mehr den Erwartungen entsprechen, die neuen Eckpunkte können dazu genutzt werden, daß die alten Eckpunkte exakt interpoliert werden. Man wählt also

$$\begin{aligned} b_{i,-1} = b_{i0}, \quad b_{i,n+1} = b_{in} \quad &\text{für } 0 \le i \le m \\ b_{-1,j} = b_{0j}, \quad b_{m+1,j} = b_{mj} \quad &\text{für } 0 \le j \le n \end{aligned}$$

für die Randpunkte. Um die Eckpunkte zu interpolieren muß gelten:

$$s(u_0, v_0) = b_{00}, \quad s(u_m, v_0) = b_{m0}, \quad s(u_0, v_n) = b_{0n}, \quad s(u_m, v_n) = b_{mn}$$

Nun wissen wir jedoch, wie sich der Kontrollpunkt zu einem Parameterpunkt berechnet und wir erhalten:

$$\begin{aligned} s(u_0, v_0) &= \frac{1}{36} \left(b_{-1,-1} + b_{-1,1} + b_{1,-1} + b_{11}\right) + \\ &\quad \frac{1}{9} \left(b_{-1,0} + b_{0,-1} + b_{01} + b_{10}\right) + \frac{4}{9} \, b_{00} \\ &= \frac{1}{36} \left(b_{-1,-1} + b_{01} + b_{10} + b_{11}\right) + \\ &\quad \frac{1}{9} \left(b_{00} + b_{00} + b_{01} + b_{10}\right) + \frac{4}{9} \, b_{00} \end{aligned}$$

Die Bedingungen für die Eckpunkte lauten also

$$\begin{aligned} b_{-1,-1} &= 12 \, b_{00} - 5 \, b_{01} - 5 \, b_{10} - b_{11} \\ b_{-1,n+1} &= 12 \, b_{0n} - 5 \, b_{0,n-1} - 5 \, b_{1n} - b_{1,n-1} \\ b_{m+1,-1} &= 12 \, b_{m0} - 5 \, b_{m-1,0} - 5 \, b_{m1} - b_{m-1,1} \\ b_{m+1,n+1} &= 12 \, b_{mn} - 5 \, b_{m-1,n} - 5 \, b_{m,n-1} - b_{m-1,n-1} \end{aligned}$$

Phantompunkte Will man, daß alle Randpunkte interpoliert werden, verwendet man am besten Phantompunkte, d.h. man sucht $2(n+1+m+1)+4$ Phantompunkte, die das bisherige Kontrollpunkt–Netz umgeben. Das Auffinden erfolgt wiederum durch Lösen eines linearen Gleichungssystems. Die Forderungen sind einerseits:

$$\begin{array}{llll} s(u_i,v_0)=b_{i0}, & s(u_i,v_n)=b_{in} & \text{für } 0\le i\le m \\ s(u_0,v_j)=b_{0j}, & s(u_m,v_j)=b_{mj} & \text{für } 0<j<n \end{array}$$

was zu $2(m+n)$ Gleichungen führt. Als Beispiel erhält man für $s(u_i,v_0)=b_{i0}$:

$$\begin{aligned} &\frac{1}{36}b_{i-1,-1}+\frac{1}{9}b_{i,-1}+\frac{1}{36}b_{i+1,-1}= \\ &\quad \frac{5}{9}b_{i0}-\frac{1}{36}b_{i-1,1}-\frac{1}{36}b_{i+1,1}-\frac{1}{9}b_{i-1,0}-\frac{1}{9}b_{i,1}-\frac{1}{9}b_{i+1,0} \end{aligned}$$

Es fehlen also andererseits noch acht Bedingungen, die man durch Anforderungen an die Steigung in den Eckpunkten aufstellen kann. Wir legen z.B. fest, daß die Richtungsvektoren der u–Ableitungen und der v–Ableitungen für die Eckpunkte in dem Kontrollpunkt–Netz liegen, d.h. daß gilt:

$$\begin{aligned} \frac{\partial s}{\partial u}(u_0,v_0) &= m(b_{10}-b_{00}) \\ -\frac{\partial s}{\partial u}(u_m,v_0) &= m(b_{m-1,0}-b_{m0}) \\ \frac{\partial s}{\partial u}(u_0,v_n) &= m(b_{1n}-b_{0n}) \\ -\frac{\partial s}{\partial u}(u_m,v_n) &= m(b_{m-1,n}-b_{m,n}) \\ \\ \frac{\partial s}{\partial v}(u_0,v_0) &= n(b_{01}-b_{00}) \\ \frac{\partial s}{\partial v}(u_m,v_0) &= n(b_{m1}-b_{m0}) \\ -\frac{\partial s}{\partial v}(u_0,v_n) &= n(b_{0,n-1}-b_{0n}) \\ -\frac{\partial s}{\partial v}(u_m,v_n) &= n(b_{m,n-1}-b_{mn}) \end{aligned}$$

Wegen

$$\frac{\partial s}{\partial u}(u,v) = \sum_{i=-1}^{m+1}\sum_{j=-1}^{n+1} b_{ij}\frac{dN_i^4}{du}(u+\frac{2}{x})N_j^4(v+\frac{2}{y})$$

ergibt sich für den Richtungsvektor an einem zu einem Kontrollpunkt b_{ab} gehörenden Parameterpunkt

$$\begin{aligned} \frac{\partial s}{\partial u}(u_a,v_b) = &\left[\frac{dN_{a-1}^4}{du}(2)\ \ \frac{dN_a^4}{du}(2)\ \ \frac{dN_{a+1}^4}{du}(2)\right] \\ &\begin{bmatrix} b_{a-1,b-1} & b_{a-1,b} & b_{a-1,b+1} \\ b_{a,b-1} & b_{a,b} & b_{a,b+1} \\ b_{a+1,b-1} & b_{a+1,b} & b_{a+1,b+1} \end{bmatrix} \begin{bmatrix} N_{b-1}^4(2) \\ N_b^4(2) \\ N_{b+1}^4(2) \end{bmatrix} \end{aligned}$$

was durch Berechnen der B–Spline–Funktionen und deren Ableitungen (siehe entsprechendes Kapitel) zu

$$\begin{aligned}\frac{\partial s}{\partial u}(u_a, v_b) &= m\begin{bmatrix}-\frac{1}{2} & 0 & \frac{1}{2}\end{bmatrix}\begin{bmatrix} b_{a-1,b-1} & b_{a-1,b} & b_{a-1,b+1} \\ b_{a,b-1} & b_{a,b} & b_{a,b+1} \\ b_{a+1,b-1} & b_{a+1,b} & b_{a+1,b+1}\end{bmatrix}\begin{bmatrix}\frac{1}{6} \\ \frac{2}{3} \\ \frac{1}{6}\end{bmatrix} \\ &= m\Big(-\frac{1}{12}b_{a-1,b-1} - \frac{1}{3}b_{a-1,b} - \frac{1}{12}b_{a-1,b+1} \\ &\qquad +\frac{1}{12}b_{a+1,b-1} + \frac{1}{3}b_{a+1,b} + \frac{1}{12}b_{a+1,b+1}\Big)\end{aligned}$$

führt. In analoger Weise erhält man

$$\begin{aligned}\frac{\partial s}{\partial v}(u_a, v_b) &= n\begin{bmatrix}\frac{1}{6} & \frac{2}{3} & \frac{1}{6}\end{bmatrix}\begin{bmatrix} b_{a-1,b-1} & b_{a-1,b} & b_{a-1,b+1} \\ b_{a,b-1} & b_{a,b} & b_{a,b+1} \\ b_{a+1,b-1} & b_{a+1,b} & b_{a+1,b+1}\end{bmatrix}\begin{bmatrix}-\frac{1}{2} \\ 0 \\ \frac{1}{2}\end{bmatrix} \\ &= n\Big(-\frac{1}{12}b_{a-1,b-1} + \frac{1}{12}b_{a-1,b+1} - \frac{1}{3}b_{a,b-1} \\ &\qquad +\frac{1}{3}b_{a,b+1} - \frac{1}{12}b_{a+1,b-1} + \frac{1}{12}b_{a+1,b+1}\Big)\end{aligned}$$

Die Forderung $\frac{\partial s}{\partial u}(u_0, v_0) = m(b_{10} - b_{00})$ wird damit beispielsweise zur Gleichung

$$\frac{1}{12}b_{-1,-1} + \frac{1}{3}b_{-1,0} + \frac{1}{12}b_{-1,1} - \frac{1}{12}b_{1,-1} = b_{00} - \frac{2}{3}b_{1,0} + \frac{1}{12}b_{1,1}$$

B–Spline–Oberflächen verlaufen näher an dem sie definieren Kontrollnetz, d.h. sie sind nicht so glatt wie entsprechende BÉZIER–Oberflächen.

4.5.4 Geschlossene B–Spline–Oberflächen

Analog zu geschlossenen B–Spline–Kurven kann man geschlossene B–Spline–Flächen durch periodische Fortsetzung der Kontrollpunkte erhalten.

4.5.5 Interpolierende B–Spline–Oberflächen

Auch das Interpolieren einer Kontrollpunktmenge ist mit dem bereits für Kurven vorgestellten Verfahren möglich. Möchte man jedoch auch hier den Rand kontrollieren, muß man wiederum $2(m+1) + 2(n+1)$ Phantompunkte einführen, die die Steigung in den Randpunkten, die interpoliert werden sollen, bestimmen.

Hier endet die Vorlesung. Für den interessierten Leser sei auf die angegebene Literatur verwiesen. Insbesondere auf die Definition mittels NURBS (nonuniform–rational–B–Spline), die inzwischen als weitverbreiteter Standard in der CAD–Welt gelten.