Modellierung graphischer Simulationen am PC

Inhaltsverzeichnis:

Rendern = Erstellen eines digitalen Bildes aus vorhandenen Daten

Raytracing = Strahlenverfolgung

SceneGraph = *SzenenGraph* = Graph aller Objekte in der aktuellen Szene

Shading = Schattierungsmodell

Primärstrahl = Strahl der vom Betrachterpunkt losgeschickt wird

Sekundärstrahl = Strahl der von einem von einem Primärstrahl (oder Sekundärstrahl) getroffenen Objekt losgeschickt wird

Ambiente Komponente = von allen Seiten einfallendes Lichtkomponente (zur Herstellung einer gewissen Stimmung in Filmszenen genutzt)

Diffuse Komponente = durch Lichtstreuung berechnete Komponente

Spekulare Komponente = durch spekulare/direkte Reflexion berechnete Komponente

0 Zielsetzung der Facharbeit

Das Thema der Facharbeit lautet 'Modellierung graphischer Simulationen am PC' und ist damit trotz des klangvollen Titels recht weit gehalten. Deshalb muss der Fokus dieser Arbeit im Folgenden weiter gebündelt werden.

Man kann graphische Simulationen am PC grob in 2 Typen unterteilen, die sich aber in den letzten Jahren durch den technischen Fortschritt auch immer näher gekommen sind:

- photo-realistische Simulationen
- Echtzeitsimulationen

Während bei ersterem Darstellungsqualität und Darstellungsmöglichkeiten das Hauptaugenmerk ausmachen, überwiegt bei letzterem – wie der Name schon sagt – der Wunsch nach größtmöglicher Interaktivität und Darstellungsgeschwindigkeit.

Wie der Leser sicher schon erraten hat, sind Computerspiele, 3D-Anwendungen wie CAD-Applikationen, etc. Echtzeitsimulationen und Programme, die z.B. von Pixar oder Disney zum Erstellen von Animationsfilmen oder von Architekten zur Erstellung von Präsentationen verwendet werden, photo-realistische Simulationen.

Die Facharbeit wird auf photo-realistische Simulationen, genauer gesagt die Untergruppe der klassischen Raycaster (der Begriff wird im nächsten Abschnitt erklärt), und die mathematischen Aufgabenstellungen, die dabei auftreten, näher eingehen.

Die Entwicklungen der Algorithmen in den letzten Jahren und die dadurch ermöglichten Optimierungen und Beschleunigungen der Darstellungsprozesse sprengen den schulischen Wissensrahmen und damit auch den Rahmen dieser Facharbeit, deswegen werde ich nur die klassischen, ursprünglichsten und notwendigsten Algorithmen und Anschauungsweisen am Beispiel eines selbstgeschriebenen *Raytracers* betrachten und später ausgewählte Probleme , die während der Entwicklung auftraten, im Detail untersuchen und ihre Lösung erläutern.

1 Was ist ein "*Raytracer*"?

Photo-realistische Simulationen versuchen logischerweise meistens Photorealismus durch eine möglichst gute Approximation physikalischer Begebenheiten zu erreichen. **Raytracer**, was auf Deutsch so viel wie **Strahlenverfolgunger** bedeutet, beruhen dabei auf Strahlenverfolgung (wer hät's gedacht). Illustration 1.1 verdeutlicht dabei das Prinzip: Vom virtuellen Betrachter werden Lichtstrahlen ausgesendet und der Auftreffpunkt mit den Objekten im (virtuellen)

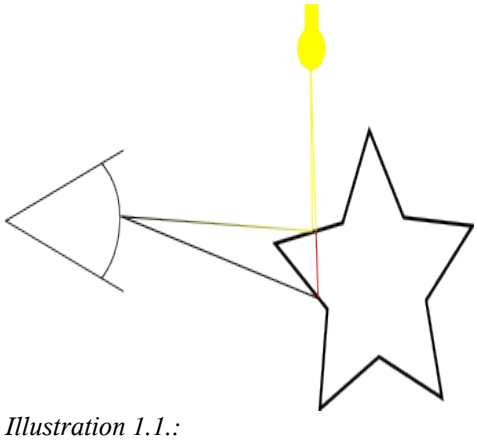

Illustration 1.1.:

Raum bestimmt. Von diesem Auftreffpunkt werden dann sogenannte **Sekundärstrahlen** losgeschickt um die Farbe des Objekts an dieser Stelle zu bestimmen. Diese können zum Beispiel die zu reflektierende Farbe bestimmen oder zur Ermittlung der Einfallsstärke des Lichtes von Lichtern (wie in der Illustration dargestellt) benutzt werden. Die Darstellung von Schatten ist damit einfach zu realisieren.

Dadurch lassen sich photorealistische Bilder durch wenige, einfache Algorithmen *rendern.*

2 Überblick über die verschiedenen Teilbereiche und den Gesamtaufbau

Folgendes Schema verdeutlicht das gedankliche Model und die verschiedenen Objekttypen, die in einem einfachen Raytracer Verwendung finden:

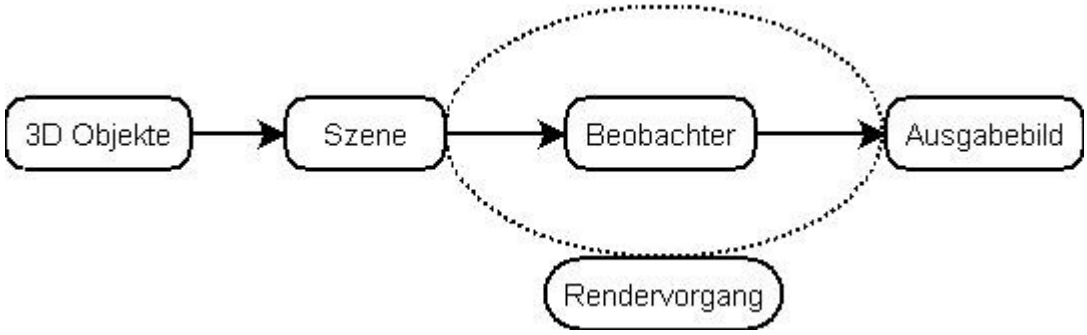

Illustration 2.1.: einfaches Schema eines Raytracers

Es müssen folgende Aufgaben bewältigt werden um ein Bild zu *rendern*:

- Verwalten einer hierarchischen Szene voller Objekten durch einen Szenen-Graph
- Transformation der Objekte vom lokalen Objektkoordinatenraum in den globalen Szenenkoordinatenraum
- Erstellen von **Primärstrahlen**, deren später bestimmter Farbwert dann einem Punkt im Ausgangsbild zugeordnet wird
- Verfolgung von Strahlen und Ermittlung des ersten Kontaktpunktes mit einem Objekt, sowie seiner Normalen und anderer wichtiger Eigenschaften
- Ermittlung des Farbwertes eines Kontaktpunktes durch Berücksichtigung von Lichtern in der Szene und gegebenenfalls Reflexionen und Refraktionen (Lichtbrechungen)

3 Nähere Betrachtung und Untersuchung einzelner Teilaspekte

3.1 Computer-bedingte Beschränkungen und Schwierigkeiten

Während in der Mathematik alles exakt berechnet werden kann, ist dies bei

einem Computer nicht der Fall. Er kann Zahlen nur bis zu einer gewissen Genauigkeit speichern und dadurch ergeben sich einige Komplikationen bei der Berechnung und dem Vergleich von Werten.

Zuerst soll jedoch das Format, in dem Zahlen in einem Computer gespeichert werden, erläutert werden. Obwohl es viele verschiedene Arten gibt Daten binär zu speichern, werden gewöhnlich zwei standardisierte IEEE-Formate[IEEE 754] benutzt um ganze Zahlen *(integers)* und Fließkommazahlen *(floating-point numbers – kur: floats)* zu speichern. Ein Computer speichert Daten unterteilt in *Bytes*, welche wiederum aus 8 *Bits* bestehen, die entweder gesetzt*(1)* oder gelöscht*(0)* sind. Ein einfacher *2-byte integer ohne Vorzeichen* (unsigned integer) könnte wie folgt dargestellt werden:

0	***1***	***0***	***0***	***1***	***1***	***1***	***0***	***0***	***0***	***0***	***1***	***0***	***0***	***1***	***1***
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Tabelle 3.1: unsigned 2-byte integer

Der Wert würde sich aus $\sum_{i=0}^{15} 2^i b_i$ ergeben wobei b_i den Wert des Bits an i-ter stelle beschreibt (siehe Tabelle 3.1). In unserem Beispiel wäre also der Wert dieser Ganzzahl: $1+2+16+512+1024+2048+16384=19987$

Bei vorzeichenbehafteten Zahlen *(signed integers)* wird das höchste Bit *(most significant bit – MSB)* benutzt um das Vorzeichen zu speichern: 0 für positive Zahlen, 1 für negative. Bei negativen Zahlen enthalten die restlichen Datenbits das 2-Komplement des absoluten Zahlenwerts – dies erleichtert die Rechenoperationen, dies sei jedoch nur am Rande bemerkt, da eine weitergehende Behandlung von Integern für einen Raytracer nicht notwendig ist.

Es soll damit lediglich ein Überblick vermittelt werden um den Einstieg in *floats* zu vereinfachen. Ein *float* setzt sich immer aus einer Mantisse, einem Exponenten und einem Vorzeichenbit, das im *MSB* gespeichert wird, zusammen:

Vorzeichen	Exponent	Mantisse

Tabelle 3.2: Struktur eines Floats

Dabei ergibt sich der resultierende Zahlenwert durch $m \cdot 2^{\exp - BIAS} \cdot \begin{cases} 1 \text{ für s} = 0 \\ -1 \text{ für s} = 1 \end{cases}$ mit **s** als Wert des Vorzeichenbits, **exp** als Exponent, der als vorzeichenloser Integer gespeichert wird, und **m** als Mantisse, die als Fixpunktzahl der Form $b.bbbb\ldots$ gespeichert wird. Dabei wird die Einerziffer immer implizit als 1 angenommen. Vom Exponenten wird weiterhin eine Konstante **BIAS** abgezogen um auch negative Exponenten zu ermöglichen.

Floats kommen in mehreren Varianten vor, die sich dadurch auszeichnen, dass sie unterschiedlich viel Datenplatz beanspruchen und damit eine unterschiedliche Genauigkeit besitzen. Mein selbst-entwickelter Raytracer verwendet sog. *"doubles"* (floats mit doppelter Genauigkeit – daher auch der

Name) für alle Operationen.

Doubles nehmen insgesamt 8 Bytes, also 64 Bits, in Anspruch, daraus werden 52 Bits für die Mantisse und 11 Bits für den Exponenten verwendet. D.h. mit einem Bias von $+1023$, können, da 0 und 2047 als Exponenten nicht erlaubt sind, Werte von $\pm 1.0 * 2^{(1-1023)} \approx \pm 2.22\text{e-}308$ bis $\pm 2.0 * 2^{(2046-1023)} \approx \pm 1.80\text{e}308$ dargestellt werden. Wichtiger ist jedoch, dass eine Genauigkeit von $\log_{10} 2^{53} \approx 15.95$, also 15 Stellen, erreicht wird (53, da die Einerziffer implizit vorausgesetzt wird und deshalb nicht gespeichert werden muss und man dadurch ein Datenbit mehr gewinnt). 15 Stellen sind, wenn man es sich genau überlegt nicht besonders viel und tatsächlich ergeben sich durch diese ungenaue Darstellung von Zahlen einige Schwierigkeiten, die schon aus Messexperimenten in Physik bekannt sind und bei der Entwicklung von graphischen Simulationen eine gewisse Sorgfalt im Detail voraussetzen:

- allgemein sind Ergebnisse von Operationen, die wegen Äquivalenz der gleich sein sollten, nicht gleich. Als Beispiel berechnen wir die häufig in genutzte Konstante ϵ für die gilt $\epsilon = inf\{x | 1 + x > 1\}$: Der Exponent von 1 ist logischerweise gleich 0, d.h. wir suchen die kleinste Zahl, die die Mantisse noch verändern kann, also $\epsilon = 2^{-53} \approx 1.1\text{e-}16$. Diese Zahl ist relativ wichtig, da man, statt direkt auf Gleichheit zu prüfen, besser prüft, ob, für ein im Rahmen der Messungenauigkeit selbstgewähltes $E > \epsilon$, gilt: $|a - b| < E$. Am Besten stellt es sich jedoch heraus den relativen Fehler zu überprüfen: $\left| \frac{a-b}{max(a,b)} \right| < \epsilon$
- die Assoziativgesetze gelten nicht.

  Beispiel: $a := \frac{\epsilon}{2}$, dann ist $(1+a)+a = 1+a = 1$, aber $1+(a+a) = 1+\epsilon \neq 1$

  D.h. es ist sehr wichtig möglichst vereinfachte Formeln zu verwenden, also Formeln mit möglichst wenigen Operationen und möglichst vielen Werten der gleichen Größenordnung (, um möglichst viele Stellen der Mantisse zu nutzen).

3.2 Koordinatensysteme und -transformationen

3.2.1 Wechsel von Koordinatenräumen

Wie schon erwähnt, werden Objekte in einem lokalen Koordinatenraum gespeichert, der es ermöglicht die räumlichen Beziehungen – Ebenen, Punkte, Oberflächen – unabhängig von der jeweiligen, aktuellen Positionierung und Drehung des Objekts in der Szene zu beschreiben. Dabei wird ein kartesisches Koordinatensysten (mit orthonormaler Basis) für alle Koordinatenräume zugrunde gelegt. Transformationen zwischen den Räumen werden dabei mit Hilfe von Matrizen vorgenommen [LEN01].
Wie man sich leicht herleiten kann, kann man folgende 3x3 Matrix benutzen um von einer Basis B zu einer anderen orthonormalen Basis $B' = \left\{ \vec{b}_1, \vec{b}_2, \vec{b}_3 \right\}$ zu wechseln (, wobei die neuen Basisvektoren im Koordinatenraum von B

angegeben sind) : $\vec{v}_{B'}=\left[\vec{b}_1\vec{b}_2\vec{b}_3\right]^{-1}\vec{v}_B=\left[\vec{b}_1\vec{b}_2\vec{b}_3\right]^{T}\vec{v}_B=\begin{pmatrix}\vec{b}_1\cdot\vec{v}_B\\ \vec{b}_2\cdot\vec{v}_B\\ \vec{b}_3\cdot\vec{v}_B\end{pmatrix}$

Beweis:

$$\vec{v}_{B'}=\left[\vec{b}_1\vec{b}_2\vec{b}_3\right]^{-1}\vec{v}_B\Leftrightarrow\left[\vec{b}_1\vec{b}_2\vec{b}_3\right]\vec{v}_{B'}=\left[\vec{b}_1\vec{b}_2\vec{b}_3\right]\left[\vec{b}_1\vec{b}_2\vec{b}_3\right]^{-1}\vec{v}_B\Leftrightarrow\left[\vec{b}_1\vec{b}_2\vec{b}_3\right]\vec{v}_{B'}=\vec{v}_B$$

Die letzte Gleichung ist nichts anderes als der Ansatz, den wir im Unterricht zum Basiswechsel benutzen. #

(Für den Beweis, dass die Inverse einer aus solchen Basisvektoren zusammengesetzten 3x3 Matrix deren Transponierte ist, siehe *3.2.3 Inversion von orthonormalen 3x3 und speziellen 4x4 Matrizen*)

Eine andere, gleichermaßen hilfreiche Anschauungsweise besteht darin, die neue Basis, durch die Repräsentation der alten Basis durch sie auszudrücken, aber dies sei hier nur am Rande erwähnt.

3.2.2 Homogene Koordinaten und Translationen

3x3 Matrizen eigenen sich zwar für alle möglichen Transformationen, aber eins vermögen sie nicht: Translationen bzw. Koordinatenursprungswechsel.
Um dies zu bewerkstelligen braucht man 4x4 Matrizen und homogene Koordinaten. Während normale Vektoren Elemente des $\mathbb{R}^3$ sind, sind homogene Vektoren und ihre Koordinaten im $\mathbb{R}^4$ angegeben: $\vec{v}:=\langle x,y,z,w\rangle\in\mathbb{R}^4$.

Homogene Vektoren können für $w\neq 0$ durch $\vec{v}\,'=\langle\frac{x}{w},\frac{y}{w},\frac{z}{w}\rangle$ oder für $w=0$ einfach durch Weglassen der w-Koordinate in normale Vektoren umgewandelt werden. Allgemein kann man sich die Umwandlung von homogenen Vektoren in Vektoren des $\mathbb{R}^3$ als Projektion jener in die Hyperebene/den Raum mit Normale $\langle 0,0,0,1\rangle$ und Abstand 1 zum Ursprung vorstellen[1] bzw. für Vektoren mit der w-Komponente 0 als Richtung des Vektors hin ins Unendliche.[MW01]

Betrachten wir nun eine Translationsmatrix, die einen homogenen Vektor $\vec{p}=\langle x,y,z,w\rangle$ um einen Richtungsvektor $\vec{T}=\langle T_x,T_y,T_z\rangle$ verschiebt:

$$\vec{p}\,'=\begin{bmatrix}1&0&0&T_x\\0&1&0&T_y\\0&0&1&T_z\\0&0&0&1\end{bmatrix}\vec{p}=\begin{pmatrix}1\cdot x+T_x\cdot w\\1\cdot y+T_y\cdot w\\1\cdot z+T_z\cdot w\\w\end{pmatrix}=\vec{p}+\begin{pmatrix}T_x\cdot w\\T_y\cdot w\\T_z\cdot w\\w\end{pmatrix}$$

Wie man sieht, fällt die Verschiebung für $w=0$ weg und wird ansonsten für $w\neq 0$ um den Faktor w vervielfacht. Dies lässt eine interessante und hilfreiche Interpretation der w-Komponente zu:
Richtungsvektoren des $\mathbb{R}^3$ werden in homogene Vektoren durch $\vec{d}=\langle x,y,z,0\rangle$ umgewandelt und werden dadurch von Translationen nicht betroffen.

---

1 Eine Projektion auf eine Gerade im $\mathbb{R}^2$ oder eine Ebene im $\mathbb{R}^3$ kann durch $\vec{p}\,'=\frac{d}{\vec{n}\cdot\vec{p}}\,\vec{p}$ beschrieben werden. d ist dabei der Abstand zum Ursprung und $\vec{n}$ die Ebenennormale. Für den $\mathbb{R}^4$-Raum kann dann der Analogieschluss vollzogen werden.

Ortsvektoren des $\mathbb{R}^3$ werden durch $\vec{d}=\langle w\cdot x, w\cdot y, w\cdot z, w\rangle$ umgewandelt, wobei für w der Einfachheit halber 1 angenommen wird.
Computeranwendungen nehmen für homogene Ortsvektoren meistens implizit 1 für w an um leichter rechnen zu können.

3.2.3 Inversion von orthonormalen 3x3 und speziellen 4x4 Matrizen

Normalerweise muss man, um Matrizen zu invertieren, auf Verfahren wie das Gauss-Verfahren zurückgreifen. Da in meinem Raytracer nur orthonormale Basen und damit orthogonale Matrizen Verwendung finden, kann jedoch auf einen „Trick" zurückgegriffen werden: $M:=\left[\vec{m}_1\vec{m}_2\vec{m}_3\right];\left(M^T\cdot M\right)_{ij}=\vec{m}_i\cdot\vec{m}_j$
Da es sich um eine orthonormale Basis handelt, ist $\vec{m}_i\cdot\vec{m}_j=0$ für $i\neq j$ und $\vec{m}_i\cdot\vec{m}_j=1$ für $i=j$, d.h. bei dem Ergebnis der Multiplikation handelt es sich um die Einheitsmatrix und damit gilt: $M^{-1}=M^T$. #

Das Invertieren von 4x4 Matrizen ist dagegen schon schwieriger, aber auch hier ist es hilfreich einige Beschränkungen anzunehmen. So soll die 4. Zeile immer aus $0,0,0,1$ bestehen und die 4. Spalte einen homogenen Verschiebungsvektor bezeichnen. D.h. man kann die Multiplikation einer 4x4 Matrix mit einem homogenen Vektor gleichsetzen mit:

$$F\cdot\vec{p}=\begin{bmatrix} M & \vec{T} \\ 000 & 1 \end{bmatrix}\cdot\begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix}=\begin{pmatrix} M\cdot\langle x,y,z\rangle+w\vec{T} \\ w \end{pmatrix}$$

Entsprechend gilt dann für die Inverse:

$$F^{-1}\cdot F\cdot\vec{p}=\begin{bmatrix} M^{-1} & -M^{-1}\vec{T} \\ 000 & 1 \end{bmatrix}\cdot F\cdot\vec{p}=\begin{bmatrix} M^{-1} & -M^{-1}\vec{T} \\ 000 & 1 \end{bmatrix}\cdot\begin{pmatrix} M\cdot\langle x,y,z\rangle+w\vec{T} \\ w \end{pmatrix}=$$
$$=\begin{pmatrix} M^{-1}\left(M\cdot\langle x,y,z\rangle+w\vec{T}\right)-wM^{-1}\vec{T} \\ w \end{pmatrix}=\begin{pmatrix} M^{-1}M\cdot\langle x,y,z\rangle+wM^{-1}\vec{T}-wM^{-1}\vec{T} \\ w \end{pmatrix}=$$
$$=\begin{pmatrix} \langle x,y,z\rangle \\ w \end{pmatrix}=\vec{p}$$

#

Mit Hilfe dieser beiden Ansätze lassen sich die Matrizeninversionen des Raytracers stark vereinfachen und beschleunigen und komplizierte Verfahren werden nicht benötigt.

3.2.4 Transformation von Oberflächennormalen

Ein Problem, dass sich noch ergibt, ist die Transformation von Oberflächennormalen. Intuitiv denkt man zwar, dass Normalen genauso transformiert werden sollten wie andere Vektoren, aber folgende Überlegung zeigt, dass dem nicht so ist:

$$\vec{v_B}\cdot\vec{n_B}=0\wedge\vec{v_{B'}}\cdot\vec{n_{B'}}=0\wedge\vec{v_{B'}}=M\vec{v_B}$$
$$\Rightarrow\vec{v_{B'}}\cdot\vec{n_{B'}}=\vec{v_{B'}}^T\vec{n_{B'}}=(M\vec{v_B})^T\vec{n_{B'}}=\vec{v_B}^TM^T\vec{n_{B'}}=0=\vec{v_B}^TM^T\left(M^T\right)^{-1}\vec{n_B}=\vec{v_B}^T\vec{n_B}=\vec{v_B}\cdot\vec{n_B}$$
$$\Rightarrow\vec{n_{B'}}=\left(M^T\right)^{-1}\vec{n_B}$$

Wie man sieht, muss man einen Normalenvektor mit der inversen

Transponierten multiplizieren. Bei einer orthogonalen Matrix (also einer Transformation zu einer orthonormalen Basis) ist ja gerade die Transponierte die Inverse und man multipliziert damit mit der inversen Inversen, also der Ursprungsmatrix. Die Intuition hat also in diesem Spezialfall recht.

3.3 Strahlenverfolgung und Kontaktpunktbestimmung

Strahlen(*Rays)* sind nichts anderes als Halbgeraden; sie können also durch 2 Eigenschaften, nämlich Ursprung und Richtung, eindeutig bestimmt werden. Die parameterisierte Funktionsform unterscheidet sich kaum von der einer normalen Geraden: $R: t \to \vec{R}(t) = \vec{s} + t\vec{d}\,; t \in \mathbb{R}_0^+$

Mit dieser Darstellung aber lassen sich alle Kontaktpunkte mit verschiedenen Objekten bestimmen. Außerdem lässt sich durch einfache Vergleiche mit der Relation zwischen dem Parameter und der Entfernung vom Ursprung des Strahles feststellen mit welchem Objekt der Strahl zuerst Kontakt hat.

3.3.1 Kontakt mit einer Ebene

Ebenen werden durch die allgemeine Gleichung dargestellt:
$P(x|y|z) \in E \Leftrightarrow Ax + By + Cz + D = 0$
Diese kann durch $Ax + By + Cz + D = \vec{n} \cdot \vec{p} - \vec{n} \cdot \vec{p}_0 = 0 \Leftrightarrow \vec{n} \cdot \vec{p} = \vec{n} \cdot \vec{p}_0$ veranschaulicht werden. Somit repräsentieren die Konstanten A, B, C die Komponenten der Ebenen-Normale und D den negierten Abstand der Ebene vom Ursprung.

Nun zur Bestimmung des Kontaktpunktes K (bzw seines Ortsvektors $\vec{k}$) eines Strahles R mit einer Ebene E mit Normale $\vec{n}$ und Abstand D vom Ursprung:

$$\vec{n} \cdot \vec{R}(t) - D = 0 \Leftrightarrow \vec{n} \cdot \left(\vec{s} + t\vec{d}\right) = D \Leftrightarrow \vec{n} \cdot \vec{s} + t\,\vec{n}\vec{d} = d \Leftrightarrow t = \frac{D - \vec{n} \cdot \vec{s}}{\vec{n} \cdot \vec{d}} \text{ für } \vec{n} \cdot \vec{d} \neq 0$$

Das Punktprodukt zwischen der Ebenennormalen und der Strahlenrichtung ist nur dann 0, wenn beide senkrecht aufeinander stehen, sprich wenn der Strahl parallel zur Ebene verläuft. Dann ist t nicht definiert, es findet also kein Kontakt statt, ansonsten muss noch überprüft werden, ob $t \geq 0$ ist; ob der Kontaktpunkt sich also auch auf der Halbgeraden/dem Strahl befindet. K bzw. $\vec{k}$ lässt sich dann durch einsetzen in $\vec{R}(t)$ finden.

3.3.2 Kontakt mit einem konvexen Polyeder

Um den Kontaktpunkt mit einem solchen Polyeder zu bestimmen, kann man nicht einfach eine Formel benutzen, sondern muss auf einen Algorithmus zurückgreifen. Dabei wird das konvexe Volumen aus n Flächen mathematisch als Schnittmenge der nicht-positiven Halbräume (also Halbraum incl. Ebene) von n Ebenen, die durch die Flächen verlaufen(und deren Normale dann logischerweise nach außen zeigt), dargestellt. Ein Punkt $P(x|y|z)$ befindet sich im nicht-positiven Halbraum einer Ebene $E(A, B, C, D)$, wenn gilt:
$Ax + By + Cz + D \leq 0$

Folgender Algorithmus liefert entweder den 1. Schnittpunkt des Strahles mit dem Polyeder zurück oder nichts, falls es keinen Kontakt gibt.

1. Wähle eine der (noch übrigen) Ebenen
2. Bestimme die Kontaktdistanz und den Kontaktpunkt des Strahles mit der

Ebene (siehe oben)

3. Falls die Kontaktdistanz nicht kleiner als die bisher „beste" ist, gehe zu Schritt 6
4. Überprüfe, ob der Punkt Teil der positiven Halbräume einer der anderen Ebenen ist. Falls zutreffend, gehe zu Schritt 6
5. Speichere die Kontaktdistanz und den Kontaktpunkt als bisher „beste" Werte
6. Gehe zurück zu Schritt 1, falls noch nicht alle Ebenen getestet wurden

Würfel sind auch Polyeder und man kann diesen Algorithmus auch für sie benutzen.

3.3.3 Kontakt mit einer Kugel

Eine Kugel kann auch durch eine der Ebenengleichung ähnliche Form (d.h. eine nicht parameterisierte Form) dargestellt werden: $\|\vec{m}-\vec{p}\|=r$ (wobei $\vec{m}$ der Ursprung der Kugel und R ihr Radius sind)
Der **erste** Kontaktpunkt lässt sich dann wieder durch Einsetzen finden:

$$\|\vec{m}-\vec{R}(t)\|=r\Leftrightarrow\|\vec{m}-\vec{s}-t\,\vec{d}\|=r\Leftrightarrow\|\underbrace{(\vec{m}-\vec{s})}_{=:\vec{o}}-t\,\vec{d}\|^2=r^2\Leftrightarrow(\vec{o}-t\,\vec{d})^2=\vec{o}^2-2t\,\vec{d}\cdot\vec{o}+t^2\vec{d}^2=r^2$$

$$\vec{d}^2t^2-2\vec{d}\cdot\vec{o}\,t+\left(\vec{o}^2-r^2\right)=0$$

$$D=4\left(\vec{d}\cdot\vec{o}\right)^2-4\vec{d}^2\vec{o}^2+4\vec{d}^2\vec{r}^2$$

$$a:=\vec{d}^2;\ b:=-2\vec{d}\cdot\vec{o};\ c:=\vec{o}^2-\vec{r}^2$$

$$t_1=\frac{-b-\sqrt{D}}{2\mathrm{a}};\ t_2=\frac{-b+\sqrt{D}}{2\mathrm{a}}$$

Für $D=0$ gibt es eine, für $D>0$ zwei mögliche Kontaktpunkte, ansonsten keinen, wie die Lösungsgleichung schön illustriert. Nur eine Lösung ist jedoch interessant und zwar die, die kleiner und positiv ist. D.h. wenn $t_1<0$, dann t_2, ansonsten ist t_1 die richtige Lösung, da $t_1<t_2$ immer gilt, weil $a>0$ ist. Wenn aber auch $t_2<0$ ist, dann gibt es gar keinen Kontaktpunkt, der auf dem Strahl R liegt. Der Kontaktpunkt bzw. sein Ortsvektor lässt sich wieder durch einsetzen von t in $\vec{R}(t)$ gewinnen.

3.3.4 Kontakt mit einem Ellipsoid oder einem Quader

Ellipsoide und Quader lassen sich beide durch verzerrte/gestreckte Kugeln oder Würfel darstellen. D.h. anstatt eine neue Formel zu finden, benutzt man einfach die oben hergeleiteten, wechselt aber in ein orthogonales Koordinatensystem, das aber nicht mehr orthonormal ist.[2] Um eine Kugel oder einen Würfel zu verzerren, kann man folgende Matrix benutzen: $S=\begin{vmatrix} a & 0 & 0 & 0\\ 0 & b & 0 & 0\\ 0 & 0 & c & 0\\ 0 & 0 & 0 & 1\end{vmatrix}$

Da jedoch im gewöhnliche Koordinatenraum die Kugel als Ellipsoid erscheint, muss man die Inverse benutzten um in den Kugelkoordinatenraum zu

2 Ich beschreibe hier einen Weg, um dies mit Matrizen zu bewerkstelligen. In meinem Raytrayer jedoch, der nur orthonormale Basen unterstützt, erreiche ich das Gleiche durch manuelles ausmultiplizieren.

gelangen: $S^{-1}=\begin{vmatrix} a^{-1} & 0 & 0 & 0 \\ 0 & b^{-1} & 0 & 0 \\ 0 & 0 & c^{-1} & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$

Daraus folgt, dass Strahlrichtung und Strahlursprung mit S^{-1} transformiert werden müssen. Dann wird der Kontaktpunkt und gegebenenfalls die Kontaktnormale bestimmt, welche daraufhin mit S bzw. S^{-1} (,da glücklicherweise $S=S^T$ und damit auch $\left(S^T\right)^{-1}=S^{-1}$,) in den normalen Koordinatenraum zurücktransformiert werden müssen.

3.4 Farbberechnung und Schattierungsmodell

Bei dem Schattierungs-/Beleuchtungsmodell handelt ist sich meistens um eine grobe Vereinfachung der physikalischen Begebenheiten. Moderne Raytracer unterstützen meistens ausgeklügelte Modelle, doch ich möchte mich auf ein sehr einfaches Modell stützen, da alles andere wohl auch zu sehr in die Physik gehen würde.

Sei C die Farbe des von einem Kontaktpunkt in Richtung das Betrachters/Strahlursprungs gesendeten Lichts, dann gilt die Gleichung:

$$C=I_{ambient}*C_{ambient}+\sum_{L=sichtbares\,Licht}\left[I_{diffuse}*C_{diffuse}(L)*M_{diffuse}(K,L)+\right.$$
$$\left.I_{specular}*C_{specular}(L)*M_{specular}(K,V,L)^{I_{\text{specular power}}}\right]$$

$C_{ambient}$ ist dabei eine Szenenkonstante, und $C_{diffuse}$ und $C_{specular}$ sind Licht-spezifisch, I_* sind Materialkonstanten, V steht für den Beobachter bzw. den Ursprung des Strahles, K für den Kontakt und auf die beiden M_* Funktionen gehe ich gleich ein.

Bisher hat es sich nur um material-, licht- oder szenen-spezifische Konstanten in der Gleichung gedreht, das Verhältnis zwischen Oberfläche, Licht und Betrachter wurde jedoch nicht berücksichtigt. Die M_* Funktionen tun genau dies:

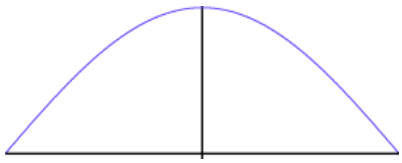

Bild 3.1.Graph des Cosinus von -90° bis 90°

$$M_{diffuse}(K,L)=\vec{n}_K\cdot\frac{(\vec{p}_L-\vec{p}_K)}{\|\vec{p}_L-\vec{p}_K\|}\text{ und}$$

$$M_{specular}(K,V,L)=\vec{r}\left(\vec{n}_K,\frac{(\vec{p}_V-\vec{p}_K)}{\|\vec{p}_V-\vec{p}_K\|}\right)\cdot\frac{(\vec{p}_L-\vec{p}_K)}{\|\vec{p}_L-\vec{p}_K\|}\ .\ \vec{n}_K$$

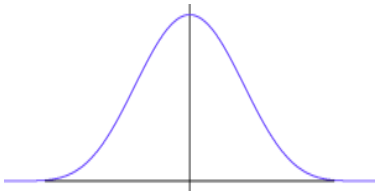

Bild 3.2. $\cos(x)^5$

$\vec{p}_*$ bezieht sich dabei auf die Kontaktnormale und auf die Position von Beobachter, Kontaktpunkt oder Licht und $\vec{r}(\vec{n},\vec{v})$ ist eine Funktion, die $\vec{v}$ an $\vec{n}$ reflektiert/spiegelt. Falls die Verwendung des Punktprodukts nicht auf Anhieb verständlich ist, so mag die alternative Form helfen: $\vec{a}\cdot\vec{b}=\|\vec{a}\|*\|\vec{b}\|*\cos\left(\sphericalangle\vec{a},\vec{b}\right)$ Wie man in Bild 3.1 sieht, nimmt $M_{diffuse}$ mit wachsendem Winkel zwischen der Normale und dem Einfallsvektor des Lichtes ab, was der Realität ziemlich nahe kommt.

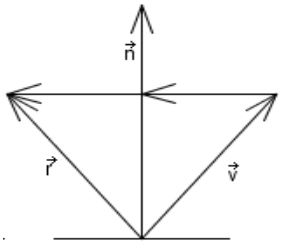

Bild 3.3.Reflexion

Gleiches gilt für $M_{specular}$ bezieht aber darin den Betrachter ein, so das für Einfallswinkel = Ausfallswinkel, die volle spekulare Lichtkomponente reflektiert wird. $I_{\text{specular power}}$ in der obersten Gleichung übernimmt dabei die Rolle der Schärferegelung, so ist bei einem Wert von 5 (wie in Bild 3.2 zusehen), der Reflexionspunkt heller aber auch begrenzter.

Als letztes sei noch $\vec{r}(\vec{n},\vec{v})$ angegeben, wobei für $\vec{n}$ und $\vec{v}$ vorausgesetzt wird, dass sie beide normalisiert sind (siehe Bild 3.3 oben für die Herleitung):
$\vec{r}(\vec{n},\vec{v}) = \vec{v} + 2\left((\vec{n}\cdot\vec{v})\vec{n} - \vec{v}\right) = \vec{v} + 2(\vec{n}\cdot\vec{v})\vec{n} - 2\vec{v} = 2(\vec{n}\cdot\vec{v})\vec{n} - \vec{v}$

Damit ist die Beschreibung dieses grundlegenden Modells fertig. Auf Reflexion, Refraktion und die Fresnel-Gleichungen verzichte ich hier aber auch, da diese „*Features*" nicht nötig sind, um den Aufbau eines *Raytracers* zu begreifen.

4 Literaturverzeichnis

[LEN01] Eric Lengyel, "Mathematics of 3D Game Programming and Computer Graphics, 2nd Edition", Hingham, USA, 2003, Charles River Media, Inc.
[IEEE 754] Floating-Point Working Group of the Microprocessor Standards Subcommittee, "IEEE Standard for BinaryFloating-Point Arithmetic", 1985, Institute of Electrical and Electronical Engineers, http://754r.ucbtest.org/standards/754.pdf, aufgerufen am 3.10.06
[MW01] Weisstein, Eric, "Homogeneous Coordinates", , , http://mathworld.wolfram.com/HomogeneousCoordinates.html, aufgerufen am 6.10.2006

6 Verwendete Software

- Microsoft „Visual Studio .NET 2003" zur Entwicklung des Raytracers
- „OpenOffice" zur Erstellung der Facharbeit
- „GIMP", „Z.u.L", „DIA" und „Powertoy Calculator" zur Erstellung der Schemata, Illustrationen, Graphen und Grafiken