LEIBNIZ-GYMNASIUM
Altdorf

Kollegstufe Abiturjahrgang 2007

Facharbeit

aus der Mathematik

Modellierung graphischer Simulationen am PC

Verfasser: ***Andreas Kirsch***

Leistungskurs: ***M2***

Kursleiter: ***Frau Rosenow***

Erzielte Punkte:
(einfache Wertung)

.....

(Unterschrift der Kursleitung)

Modellierung graphischer Simulationen am PC

Inhaltsverzeichnis:

0 Zielsetzung der Facharbeit....3
0.1 Art und Weise der Darstellung, Definition und Einführung....3
1 Was ist ein "Raytracer"?....3
2 Überblick über die verschiedenen Teilbereiche und den Gesamtaufbau....4
2.1 Szene und Szenenaufbau....4
2.2 Der Szenegraph....6
2.3 Rendervorgang....6
3 Nähere Betrachtung und Untersuchung einzelner Teilaspekte....8
3.1 Computer-bedingte Beschränkungen und Schwierigkeiten....8
3.2 Koordinatenräume und -transformationen....10
3.2.1 Koordinatenräume....10
3.2.2 Transformationen und Matrizen....11
3.2.3 Transformationen als Basiswechsel....14
3.2.4 Darstellung von Skalierungen durch Matrizen....18
3.2.5 Translationen und affine Transformationen....19
3.2.6 Umkehrung von homegenen Transformationsmatrizen....22
3.2.7 Transformation von (Ebenen)normalen....23
3.3 Strahlenverfolgung und Kontaktpunktbestimmung....23
3.3.1 Kontakt mit einer Ebene....24
3.3.2 Kontakt mit einem konvexen Polyeder....24
3.3.3 Kontakt mit einer Kugel(oberfläche)....25
3.4 Farbberechnung und Schattierungsmodell....26
4 Anhang (mit zusätzlichen, von der Korrektur ausgeschlossenen Materialien)....28
4.1 Einführung in Lineare Algebra....28
4.1.1 Vektoren und ihre Eigenschaften....28
4.1.2 Matrizen und ihre Eigenschaften....29
4.1.3 Matrizen und lineare Gleichungssysteme....32
4.2 Zusätzliche Beweise....33
4.2.1 Matrizen aus Basisvektoren sind invertierbar (für Kapitel 3.2.3 Seite 16)....33
4.3 Verwendete Software....33
4.4 Literaturverzeichnis....33
4.5 Alphabetischer Index....34

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 immer stärker angenähert haben:

- 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 (im Entwurfsmodus), 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 Raytrayer (der Begriff wird im nächsten Abschnitt erklärt) und auf die mathematischen Aufgaben- und Problemstellungen, 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, weswegen nur die klassischen, ursprünglichsten und notwendigsten Algorithmen und Anschauungsweisen beschrieben werden.

0.1 Art und Weise der Darstellung, Definition und Einführung

Da dieses Thema, insbesondere der allgemeine, programm-technische Aspekt, relativ wenig mit dem Schulstoff der gymnasialen Oberstufe zu tun hat, müssen viele neue Begriffe eingeführt, manche Definitionen (im mathematischen Teil) erweitert und die Art der Darstellung der Übersichtlichkeit wegen gelockert werden. Auch soll die Facharbeit für den uneingearbeiteten Schüler im Leistungskurs verständlich sein. Trotzdem und gerade deswegen ist es wichtig, einen einheitlichen Stil zu benutzen, um das Auffinden von Spezialbegriffen und Formeln zu erleichtern.

Deswegen werde neue Begriffe bzw. Begriffe, die ad-hoc definiert werden **fett** gedruckt. Mit ad-hoc-definierten Begriffen sind Begriffe gemeint, die im Text direkt, also beiläufig, erwähnt werden, deren Bedeutung aber trotzdem von Wichtigkeit ist. Gerade wegen der Seitenbeschränkung, die zwar für diese Facharbeit schon gelockert worden ist, kann nur für die wenigsten und wichtigsten Begriffe eine präzise Definition erfolgen. Außerdem würde eine präzise Definition bei manchen Begriffen wahrscheinlich stärker verwirren, da sie ein weitaus tieferes Verständnis für die Thematik erfordern würde.

Auf die Fähigkeit der Abstraktion des Lesers wird größter Wert gelegt. Er sei sich gewiss, dass alles für das Verständnis wichtige erwähnt und in der nötigen Tiefe behandelt wird.Zusätzlich gibt es im Anhang einen Index, der das Nachschlagen von Fachbegriffen erleichtern sollte.

1 Was ist ein "*Raytracer*"?

Photo-realistische Simulationen versuchen logischerweise Photorealismus durch eine möglichst gute Approximation physikalischer Begebenheiten zu erreichen. **Raytracer**, was auf Deutsch so viel wie **Strahlenverfolger** bedeutet, beruhen dabei auf Strahlenverfolgung. Bild 1.1 verdeutlicht dabei das Prinzip:

Vom virtuellen Betrachter werden Strahlen, sogenannte **Primärstrahlen**, ausgesandt und der Auftreffpunkt, der sogenannte **Kontaktpunkt,** mit den Objekten im (virtuellen) Raum bestimmt.

Von diesem Auftreffpunkt werden dann sogenannte **Sekundärstrahlen** losgeschickt um die Farbe des Objekts an dieser Stelle zu bestimmen.

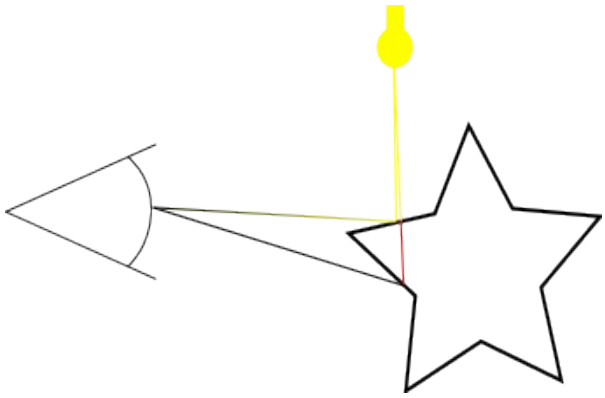

Bild 1.1.: Prinzip der Strahlenverfolgung

Diese können zum Beispiel die zu reflektierende Farbe bestimmen oder zur Ermittlung der Einfallsstärke des Lichtes von **Lichtquellen** (wie in der Illustration dargestellt) benutzt werden. Die Darstellung von Schatten ist damit einfach zu realisieren, indem geprüft wird, ob ein **Sekundärstrahl** ungehindert zu einer Lichtquelle gelangen kann oder nicht.

Mit Hilfe dieses einfachen Grundgedankens lassen sich äußert komplexe Bilder einfach **rendern**[1].

2 Überblick über die verschiedenen Teilbereiche und den Gesamtaufbau

Das Schema in Bild 2.1 verdeutlicht den gedanklichen Aufbau eines einfachen Raytracers:

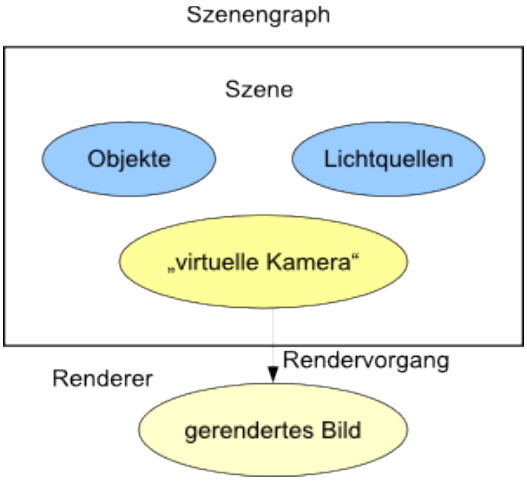

Bild 2.1.Allgemeines Schema

Wie man sieht, besteht ein Raytracer aus 2 Einheiten: dem sog. **Renderer**, der das Ausgabebild erstellt, und einem **Szenengraph**, der eine Szene auf eine nützliche Art und Weise repräsentiert, die es dem Renderer erlaubt, diese effizient zu rendern.

2.1 Szene und Szenenaufbau

Der Raytracer rendert immer eine **Szene**, welche aus **Objekten** und **Lichtquellen** besteht. Dabei dient eine **virtuelle Kamera** als unsichtbarer Beobachter. Objekte und Lichtquellen werden hierarchisch angeordnet und die Datenstruktur, welche die Verhältnisse in einer Szene beschreibt, wird **Szenengraph** genannt.

Bevor man aber den Szenengraphen genauer betrachten kann, ist es wichtig zuerst zu verstehen, was genau mit „Objekten und Lichtquellen" gemeint ist.

Lichtquellen sind Punkte im Raum, die Licht einer bestimmen Farbe in alle Richtungen aussenden und damit Objekte beleuchten. Sie haben keine Ausdehnung und sind auch nicht direkt auf einem gerenderten Bild sichtbar. Nur ihre Wirkung, die Beleuchtung anderer Objekte, ist direkt sichtbar. Lichtquellen haben Eigenschaften wie Intensität, Reichweite, Lichtfarbe (bzw. Lichtspektrum), etc.

Objekte haben dagegen einen „Körper" und sind auf dem Ausgangsbild sichtbar. Ein Objekt hat zwei verschiedene Kategorien von Eigenschaften: die eine Kategorie beschreibt die Stellung des Objektes in der Szene, die andere beschreibt sein Aussehen.

Die Art und Weise, wie das Aussehen eines bestimmten Objektes gespeichert wird, ist von Objekttyp zu Objekttyp unterschiedlich:

1 *Rendern* bezeichnet das Erstellen eines computergenerierten Bildes

Eine Kugel (wie in Bild 2.2 zu sehen) kann durch Radius und Aufhängepunkt ausreichend beschrieben werden.

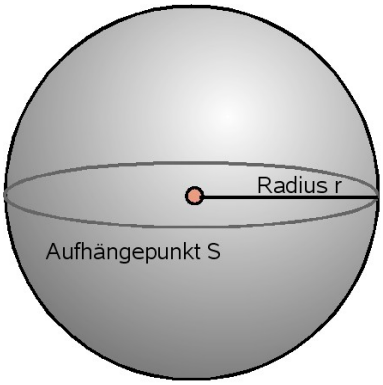

Bild 2.2.Kugel

Ein Quader dagegen kann durch Höhe, Breite und Tiefe, oder auch durch die Koordinaten seiner Ecken definiert sein (zum Vergleich Bild 2.3).

Bei parameterisierten Flächen dagegen speichert man die Funktion und die Koeffizienten (siehe Bild 2.4 für ein Beispiel einer parameterisierten Funktion).

Mit der Stellung in der Szene sind Eigenschaften gemeint wie: **Position, Ausrichtung (Drehung)** und **Größe** um nur einige zu nennen. Diese Eigenschaften sind unabhängig von dem eigentlichen Aussehen des Objektes: Ein Tisch sieht wie ein Tisch aus, egal wo im Raum er steht, wie er ausgerichtet ist oder wie groß er ist. Entsprechend ist es nützlich, das Aussehen getrennt von den nur für den Szeneaufbau wichtigen Eigenschaften, wie den oben genannten, anzugeben.

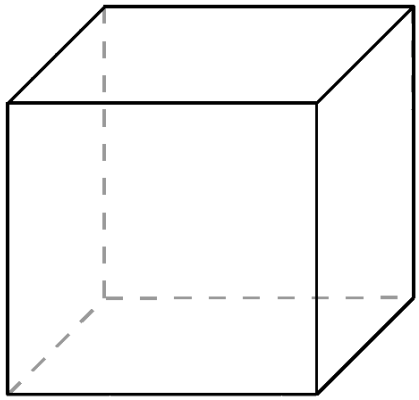
Bild 2.3.Quader

Als Beispiel stelle man sich eine Pyramide vor, deren Eckpunkte in einem kartesischen Koordinatensystem angegeben sind. Es ist nützlicher, diese Koordinaten relativ zu einander anzugeben, also **lokal**, anstatt für jede Ecke die Koordinaten relativ zur Gesamtszene (**global**) anzugeben, da, wenn zum Beispiel die Position der Pyramide geändert wird, im ersten Fall (lokal) keine einzige Koordinate direkt geändert werden müsste, während im anderen Fall (global) jede Koordiante angepasst werden müsste.

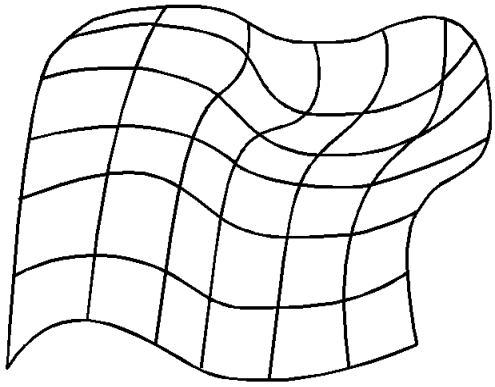
Bild 2.4.parameterisierte Oberfläche (Skizze)

Deswegen wird das Aussehen eines Objekts (z.B wie oben bei einer Pyramide die Koordinaten der Ecken) in einem **lokalen Koordinatenraum** gespeichert, der unabhängig von der jeweiligen Position, Ausrichtung und Größe des Objektes in der Szene ist. Wenn das betreffende Objekt dann in die Szene „gesetzt" wird, werden die Koordinaten, die sein Aussehen beschreiben vom lokalen in den **globalen Koordinatenraum transformiert**, d.h. das Objekt wird dann an die entsprechende Stelle in der Szene verschoben, sowie entsprechend rotiert und skaliert. In Bild 2.5 sieht man dies am Beispiel eines Kegels.

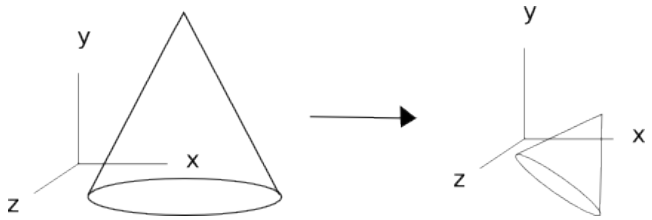

Bild 2.5.Vom lokalen in den globalen Koordinatenraum

Dieser Wechsel von einem Koordinatenraum in einen anderen wird durch sogenannte **Transformationen** durchgeführt.

Es gibt einige **Grundtransformationen**:

- **Translationen** für Verschiebungen
- **Rotationen** für Drehungen
- **Skalierungen** für Vergrößerungen/Verkleinerungen

Es ist möglich, Transformationen miteinander zu verknüpfen und so aus den Grundtransformationen komplexe Transformationen herzustellen (z.B. Skalierung->Translation->Rotation->Translation). Dies wird als **Verkettung** bezeichnet und eine

Transformation, die aus mehreren Grundtransformationen besteht, heißt auch **verkettete Transformation**.

Zusätzlich ist es möglich Grundtransformationen und verkettete Transformationen, die aus eben diesen Grundtransformationen bestehen, umzukehren. Man spricht von einer **inversen Transformation**, wenn man die Umkehrung einer Transformation meint.

2.2 Der Szenegraph

Der Szenengraph, der es, wie oben erwähnt, erlaubt, Objekte hierarchisch in der Szene zu ordnen, nutzt nun verkettete Transformationen um die Szene „aufzubauen". Was genau eine Szenengraph ist, wird am einfachsten verständlich, wenn man ein Schema vor sich sieht:

Ein leicht verständliches Beispiel ist in Bild 2.6 zu sehen. Es beschreibt eine Szene mit einer (äußeren) Hauptlichtquelle und einem Auto, das zusätzlich aus 4 verschiedenen Rädern und jeweils einem linken und rechten Scheinwerfer besteht. Wenn nun z.B. das Auto um 45° mit Hilfe einer **Rotations-Transformation** gedreht werden würde, so würde der Szenengraph dafür sorgen, dass die Räder und Scheinwerfer automatisch mitgedreht werden, da das Objekt „Auto" in der Hierarchie über den Rädern und Scheinwerfern steht. Dadurch lassen sich komplexe Szenen einfach aufbauen, da durch eine logische Anordnung der Objekte und Lichtquellen das Positionieren und das Ausrichten der Objekte vereinfacht wird. Außerdem lassen sich so komplexe Szenen leicht nachträglich bearbeiten, da man die Transformationen getrennt vom Aussehen speichern kann.

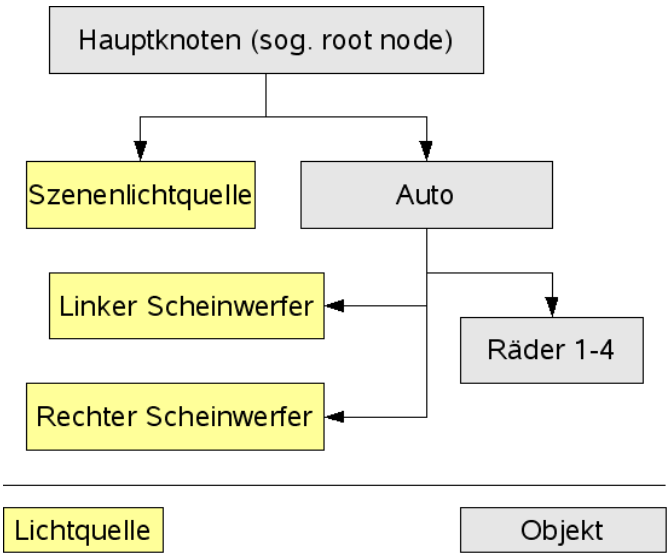

Bild 2.6.Schema eines Szenengraph für eine einfachen Szene

Man kann also zusammenfassend sagen[6]:

> Ein **Szenengraph** ist eine Baum-Struktur, die alle Objekte und Lichtquellen einer Szene hierarchisch anordnet und sie entsprechend ihrer Stellung in der Hierarchie durch **verkettete Transformationen** in den **globalen Koordinatenraum transformiert.**

Wir wissen jetzt, wie die Szene aufgebaut ist, was einen Szenengraph und was Transformationen sind.

2.3 Rendervorgang

Wie in *1 Was ist ein "Raytracer"?* beschrieben, werden in dem vereinfachten Raytracer, den ich hier beschreibe, folgende Schritte durchgeführt, um die Farbe eines Punktes im gerenderten Bild zu bestimmen (auf die Punkte wird weiter unten im Detail eingegangen):

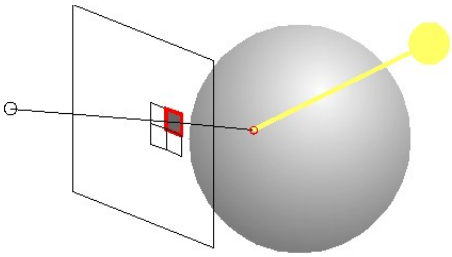
Bild 2.7.Modellvorstellung für perspektivische Projektion

1. Es wird ein **Primärstrahl** für jeden Punkt des Ausgabebildes losgeschickt. Bei perspektivischer Projektion (also normal wie bei einer Kamera), kann man es sich wie in Bild 2.7 gezeigt vorstellen: Vom Betrachtungspunkt wird ein Strahl durch den entsprechenden Bildpunkt losgeschickt.

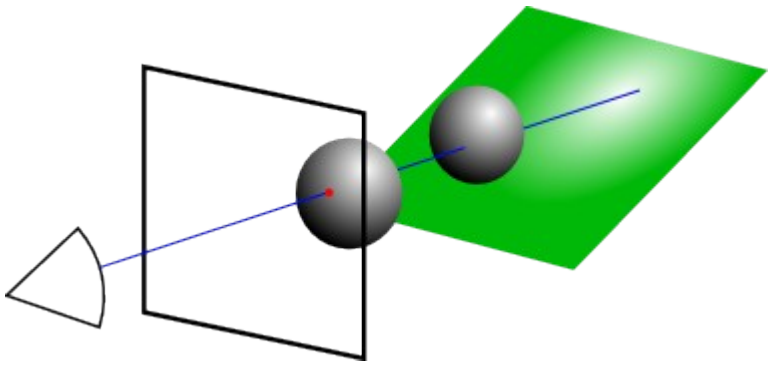

Bild 2.8.Kontaktpunktbestimmung

2. Der früheste **Kontaktpunkt** des Strahles mit einem Objekt der Szene wird bestimmt. Außerdem werden wichtige Werte, wie Entfernung zum Betrachter und Normale, der Vektor, der senktrecht auf der Kontaktoberfläche steht, bestimmt (siehe Bild 2.8).
3. Die Farbe des Objektes am Kontaktpunkt wird bestimmt und im gerenderten Bild als Farbe des jeweiligen Punktes im **Ausgabebild** gespeichert.

Zu 1.:

Zuerst einmal ist es wichtig zu definieren, was das Ausgabebild ist:

> Das **Ausgabebild** oder **gerenderte Bild** besteht aus w mal h Farbwerten (w: Breite, h: Höhe). Die Farbwerte werden gewöhnlich in ihren 3 Einzelkomponenten (rot, grün, blau) getrennt gespeichert. Dabei können die Komponenten Werte zwischen 0 und 1 annehmen. Sie bezeichnen die Intensität der jeweiligen Grundfarbe. Die Grundfarben werden dann entsprechend ihrer Intensität gemischt und ergeben durch additive Farbmischung die Farbe (vgl. Grundwissen Kunst und Physik Jahrgangsstufe 10).

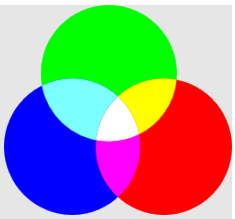

Bild 2.9.Additives Farbschema

Um zu verstehen, was genau ein **Primärstrahl** ist, muss zuerst definiert werden, was allgemein unter einem Strahl zu verstehen ist:

> Ein **Strahl** ist nichts anderes als eine Halbgerade, die durch eine Richtung und einen Startpunkt bestimmt ist.

Primärstrahlen fangen am Aufhängepunkt der virtuellen Kamera an. Ihre Richtung hängt neben der Blickrichtung der Kamera, ihrer Drehung etc., auch von der Art der **Projektion** ab.

> Die **Projektion** bestimmt, wie die virtuelle Szene auf das **Ausgabebild** abgebildet wird.

Es gibt verschiedene Projektionen:

Perspektivische Projektion:

Unter einer perspektivische Projektion versteht man eine Projektion, die einer normalen Kamera oder dem menschlichen Auge nachempfunden ist. Sprich: Objekte verjüngen sich in die Tiefe.

Ein Beispiel dafür wären parallele Gleise, die sich in der Ferne zu schneiden scheinen.

Orthogonale Projektion:

Unter einer orthogonalen Projektion versteht man eine Projektion, die längen- und winkeltreu ist (im Gegensatz zur perspektivischen Projektion, bei der die Längen mit dem Abstand zum Betrachter kleiner werden).

Ein Beispiel dafür wären Risszeichnungen und Blaupausen von Gebäuden, bei denen es auf exakte Angaben ankommt.

Zu 2.:

Wie in *2.2 Der Szenegraph* beschrieben, werden verkettete Transformationen benutzt, um Objekte vom **lokalen** in den **globalen Koordinatenraum** zu transformieren.

Tatsächlich bedient sich ein Raytracer aber normalerweise eines Tricks, um viel Arbeit zu sparen:

In Wirklichkeit transfomiert er für jedes für einen Kontakt in Frage kommende Objekt den Strahl vom **globalen** in den **lokalen Koordinatenraum**.

Um an dem Beispiel mit der Pyramide anzuknüpfen: Eine Pyramide hat mindestens 4 Ecken (Spitze&Grundfläche); es müssten also 4 Koordinaten transformiert werden. Beim Strahl dagegen müssen nur 2 Koordinaten transformiert werden: Aufhängepunkt und Richtung (wobei die Richtung anders behandelt werden muss, doch dazu später).

Nachdem der Strahl in den lokalen Koordinatenraum transformiert worden ist, wird der erste Kontaktpunkt mit dem entsprechenden Objekt (sowie die Oberflächennormale) und die Distanz vom Strahlursprung bestimmt. Kontaktpunkt, Oberflächenormale und Kontaktdistanz müssen dann vom lokalen in den globalen Koordinatenraum zurücktransformiert werden, um aussagekräftige (von den Objekten unabhängige) Daten zu erhalten.

Die „aussagekräftige" Distanz wird mit den Kontakten des Strahles mit anderen Objekten verglichen und die kleinste Distanz bestimmt den gesuchten frühsten Kontaktpunkt.

Zu 3.:

Zur Bestimmung der Farbe des Kontaktpunktes werden **Sekundärstrahlen** zu allen Lichtquellen in der Szene geschickt und es wird geprüft, ob der Sekundärstrahl vor der Lichtquelle mit einem anderen Objekt zusammentrifft. Ist dem so, so liegt der Kontaktpunkt im Schatten der betreffenden Lichtquelle und diese Lichtquelle muss bei den weiteren Berechnungen nicht weiter berücksichtigt werden.

Eine genauere Betrachtung der Farbberechnung und des Schattierungsmodelles erfolgt in *3.4 Farbberechnung und Schattierungsmodell*.

Es wurde ein grundlegender Überblick über die Art und Weise gegeben, wie das Ausgabebild erstellt wird und der Leser weiß jetzt, wie wichtig Transformationen und die Möglichkeit, Transformationen umzukehren, für den Raytracer sind.

3 Nähere Betrachtung und Untersuchung einzelner Teilaspekte

Jetzt, da der Leser einen Überblick über die Funktionsweise des Raytracers hat, ist es an der Zeit, einige Bereiche im Detail zu betrachten.

3.1 Computer-bedingte Beschränkungen und Schwierigkeiten

Es ist wichtig zu verstehen, dass Computer nicht perfekt rechnen können und es immer Rundungsfehler und Probleme mit der Genauigkeit ab einer bestimmten Größenordnung gibt.

Außerdem kann ein Computer 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[3] benutzt, um ganze Zahlen, sogenannte **Integer,** und Fließkommazahlen, sogenannte **floating-point numbers** oder kurz **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 des Integers 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 integer***)* wird das höchste Bit (**most significant bit** – oder kurz **MSB**) benutzt, um das Vorzeichen zu speichern: 0 für positive Zahlen, 1 für negative. Bei negativen Zahlen enthalten die restlichen Datenbits das 2-er 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:

Vorzeichenbit	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...* gespeichert wird. Dabei wird die Einerziffer immer implizit als 1 angenommen. Vom Exponenten wird weiterhin eine positive 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. Normalerweise verwendet man heutzutage 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 $\pm 1.0 \cdot 2^{(1-1023)} \approx \pm 2.22 \cdot 10^{-308}$ bis $\pm 2.0 \cdot 2^{(2046-1023)} \approx \pm 1.80 \cdot 10^{308}$ 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, 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 aus der

Physik bekannt sind und bei der Entwicklung von graphischen Simulationen eine gewisse Sorgfalt im Detail voraussetzen:

- allgemein sind Ergebnisse von Operationen, die wegen Äquivalenz gleich sein sollten, nicht gleich. Als Beispiel berechnen wir die häufig genutzte Konstante ϵ für die gilt $\epsilon = inf\{x | double(1+x) > double(1)\}$, wobei $double$ dabei eine Funktion bezeichnet, die eine reellen Zahl auf ihren möglicherweise gerundeten Wert als Double abbildet:

  Der Exponent von $double(1)$ ist logischerweise gleich 0, d.h. wir suchen die kleinste Zahl, die die Mantisse noch verändern kann, also $\epsilon = 2^{-53} \approx 1.1 \cdot 10^{-16}$. 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 folgt für die Berechnung von $(1+a)+a$:
  $double(double(1+a)+a) = double(1+a) = double(1)$, aber für $1+(a+a)$ gilt:
  $double(1+double(a+a)) = double(1+\epsilon) \neq 1$
  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 Koordinatenräume und -transformationen

In *2 Überblick über die verschiedenen Teilbereiche und den Gesamtaufbau* wurde schon deutlich, wie wichtig Transformationen und ihre Verkettung für einen Raytracer sind.

Im Folgenden soll ihre mathematische Darstellung genauer untersucht werden.

Transformationen werden mittels Matrizen dargestellt[4]. Da Matrizen nicht zum Schulstoff gehören, aber notwendig für das Verständnis sind, sind im Anhang (unter *4.1 Einführung in Lineare Algebra*) einige Definitionen, Gesetze, Beispiele und Aufgaben angefügt.

Auch wenn der Leser vertraut mit allem ist, so sollte er trotzdem einen Blick darauf werfen, da möglicherweise eine für ihn neue Nomenklatur eingeführt wird.

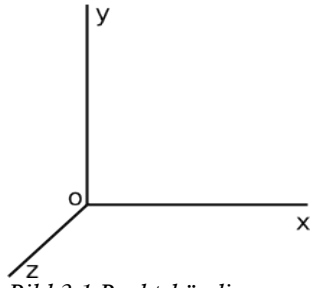

Bild 3.1.Rechtshändiges Koordinatensystem

3.2.1 Koordinatenräume

Wie in *2 Überblick über die verschiedenen Teilbereiche und den Gesamtaufbau* erwähnt, existieren ein globaler Koordinatenraum und für jedes Objekt ein eigener lokaler Koordinatenraum.

Bisher wurde nicht definiert, was genau mit dem Term **Koordinatenraum** gemeint ist.

> Ein **Koordinatenraum** besteht aus einer Basis, den dazugehörigen Basisvektoren und einem Ursprung (und bildet selbst auch einen Vektorraum [11]).

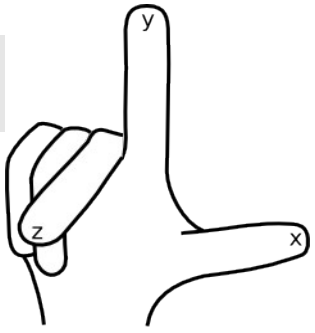

Bild 3.2.Rechte-Hand-Regel

Die Basisvektoren ordnet man dabei bestimmten Achsen in einem rechtshändigen Koordinatensystem zu. Üblicherweise werden die Achsen mit **x**, **y** und **z** bezeichnet, wobei die x-Achse dem 1., die y-Achse dem 2. und die z-Achse dem 3. Basisvektor zugeordnet wird.
Die Anordnung der 3 Achsen ergibt sich aus der **Rechten-**

Hand-Regel (siehe Bild 3.2 zur Veranschaulichung).

3.2.2 Transformationen und Matrizen

Wie in *2 Überblick über die verschiedenen Teilbereiche und den Gesamtaufbau* geschildert, dienen Transformationen dazu, von einem Koordinatenraum in einen anderen zu wechseln.

Bisher wurde die Fragestellung, was eine Transformation sei und wie sie mathematisch dargestellt wird, geschickt umgangen. Nun soll sie jedoch beantwortet werden:

> Eine **Transformation transformiert** einen Vektor von einem Koordinatenraum in einen anderen. Sie ist damit eine Funktion, die einen Vektor auf einen anderen abbildet: $T:\mathbb{R}^n\to\mathbb{R}^n$ und $\vec{x}\,'=\vec{T}(\vec{x})$ T kann auch in Komponentenschreibweise angegeben werden:
>
> $$T:\begin{pmatrix}x_1\\ \vdots\\ x_n\end{pmatrix}\to\begin{pmatrix}T_1(\vec{x})\\ \vdots\\ T_n(\vec{x})\end{pmatrix} \text{ bzw. } \vec{x}\,'=\begin{pmatrix}x'_1\\ \vdots\\ x'_n\end{pmatrix}\to\begin{pmatrix}T_1(\vec{x})\\ \vdots\\ T_n(\vec{x})\end{pmatrix}=\vec{T}(\vec{x})$$
>
> T_i ist dabei eine Funktion, die einen Vektor auf eine reelle Zahl abbildet, welche die i -te Komponente des transformierten Vektors darstellt. $T_{1,}\cdots,T_n$ werden deshalb auch als **Komponentenfunktionen** von T bezeichnet.

Beispiel:

$$T:\mathbb{R}^2\to\mathbb{R}^2$$
$$T_1(\vec{x})=x_1^3+x_2$$
$$T_2(\vec{x})=x_1^3-x_2$$
$$T:\vec{x}\to\begin{pmatrix}T_1(\vec{x})\\ T_2(\vec{x})\end{pmatrix}=\begin{pmatrix}x_1^3+x_2\\ x_1^3-x_2\end{pmatrix}$$
$$\vec{x}=\begin{pmatrix}-1\\ 1\end{pmatrix}$$
$$\Rightarrow\vec{T}(\vec{x})=\begin{pmatrix}0\\ -2\end{pmatrix}$$

T wäre eine gültige Transformation. Eine andere Transformation (Bilder aus [10]) ist als Beispiel hier zu sehen:

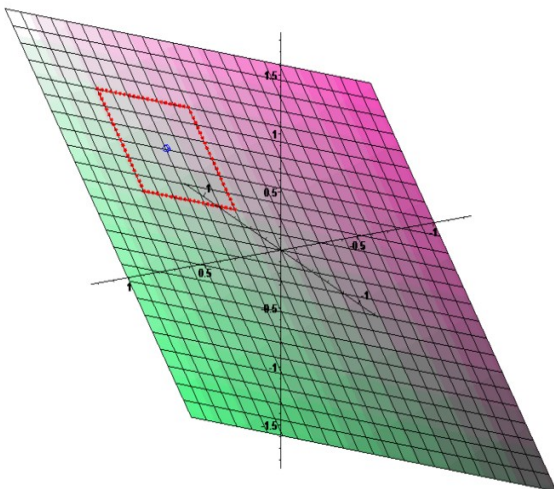

Bild 3.4.Eine Ebene vor der Transformation

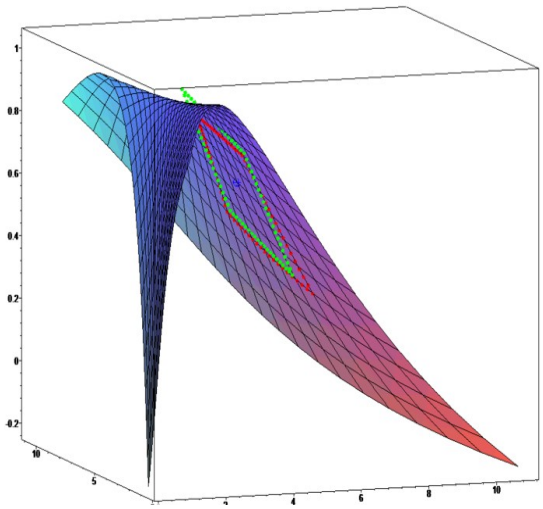

Bild 3.3.Die „Ebene“ nach der Transformation

Solche Transformationen sind aber kompliziert und zu unbestimmt, um sie genügend für den Einsatz in einem Raytracer vereinfachen zu können. Außerdem liegt das Hauptinteresse ja an der Darstellung der **Grundtransformationen** (Rotation, Translation und Skalierung) und der Möglichkeit ihrer **Verkettung**.

Deswegen wird nur eine Untergruppe der Transformationen untersucht:

> Eine **lineare Transformation** ist eine spezielle Transformation, deren Komponentenfunktionen nur aus ersten Potenzen der Vektorkomponenten bestehen. T ist genau dann eine lineare Transformation, wenn für die Komponentenfunktionen von T gilt:
>
> $$\begin{aligned} T_1(\vec{x}) &= a_{11}\cdot x_1 + a_{12}\cdot x_2 + \cdots + a_{1n}\cdot x_n \\ T_2(\vec{x}) &= a_{21}\cdot x_1 + a_{22}\cdot x_2 + \cdots + a_{2n}\cdot x_n \\ &\vdots \\ T_n(\vec{x}) &= a_{n1}\cdot x_1 + a_{n2}\cdot x_2 + \cdots + a_{nn}\cdot x_n \end{aligned}$$

Beispiele:

1. $\vec{T}(\vec{x}) = \begin{pmatrix} x_1 + 2x_3 \\ 2x_2 + 2x_3 \\ 2x_3 \end{pmatrix}$
   T ist eine lineare Transformation
2. Aber F_1 und F_2 sind keine:
   $$\vec{F}_1(\vec{x}) = \begin{pmatrix} x_1 + 3 \\ x_2 \\ x_3 \end{pmatrix};\ \vec{F}_2(\vec{x}) = \begin{pmatrix} x_1^2 \\ x_2^3 \\ x_3^4 \end{pmatrix}$$

Betrachten wir noch einmal die verschiedenen Komponentenfunktionen einer linearen Transformation T:

$$\begin{aligned} T_1(\vec{x}) &= a_{11}\cdot x_1 + a_{12}\cdot x_2 + \cdots + a_{1n}\cdot x_n \\ T_2(\vec{x}) &= a_{21}\cdot x_1 + a_{22}\cdot x_2 + \cdots + a_{2n}\cdot x_n \\ &\vdots \\ T_n(\vec{x}) &= a_{n1}\cdot x_1 + a_{n2}\cdot x_2 + \cdots + a_{nn}\cdot x_n \end{aligned}$$

Wenn man an *4.1.3 Matrizen und lineare Gleichungssysteme* denkt, fällt die Ähnlichkeit zu einem linearen Gleichungssystem auf. Schreiben wir die Transformation also einmal zum Versuch um:

$$\vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix};\ \vec{T}(\vec{x}) = \begin{bmatrix} T_1(\vec{x}) \\ T_2(\vec{x}) \\ \vdots \\ T_n(\vec{x}) \end{bmatrix};\ A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}$$

$$\left.\begin{aligned} T_1(\vec{x}) &= a_{11}\cdot x_1 + a_{12}\cdot x_2 + \cdots + a_{1n}\cdot x_n \\ T_2(\vec{x}) &= a_{21}\cdot x_1 + a_{22}\cdot x_2 + \cdots + a_{2n}\cdot x_n \\ &\vdots \\ T_n(\vec{x}) &= a_{n1}\cdot x_1 + a_{n2}\cdot x_2 + \cdots + a_{nn}\cdot x_n \end{aligned}\right\} \Leftrightarrow \vec{T}(\vec{x}) = Ax$$

(Durch Ausmultiplizieren kann dies leicht nachgeprüft werden.)

Wie man sieht, können Matrizen zur Darstellung von Transformationen äußerst nützlich sein.

Allgemein kann man feststellen:

Eine quadratische $n \times n$ Matrix stellt eine lineare Transformation für Vektoren mit n Komponenten (kurz: $\in \mathbb{R}^n$) dar. Ein Vektor wird durch Multiplikation mit der **Transformationsmatrix** transformiert: $\vec{x}' = T\vec{x}$, wobei T die Transformationsmatrix darstellt.

Das ist schön und gut, aber was ist mit den wichtigen Eigenschaften von Transformationen, deren Forderung für einen Raytracer wichtig sind:

Die Möglichkeit der Verkettung von Transformationen und dem für die Ausführung des Renderns wichtigen Vorgang der Umkehrung der Transformationen - man erinnere sich an *2.3 Rendervorgang*:
Strahlen werden vom globalen in den lokalen Koordiantenraum transformiert, also umgekehrt zur angegebenen Transformationen vom lokalen in den globalen Koordiantenraum, und die berechneten Vektoren danach vom lokalen in den globalen Koordiantenraum zurücktransformiert.

Die in *4.1.2 Matrizen und ihre Eigenschaften* aufgelisteten Sätze helfen, dies zu untersuchen:

1. **Verkettung**

   Verkettung bedeutet, dass zwei oder mehrere Transformationen zu einer einzigen zusammengefasst werden, die sich, wenn ein Vektor durch sie transformiert wird, genauso verhält, als ob die Transformationen, aus denen sie besteht, einzeln angewandt wurden.

   Das Stichwort „Matrix-Multiplikation" hilft weiter:

   Nehmen wir an, wir hätten k lineare Transformationen, die verkettet werden sollen: $L_1, \cdots, L_k$. Diese sollen dabei der Reihe nach auf einen Vektor $\vec{x}$ angewandt werden.

   Versuchen wir die Verkettung schrittweise herzuleiten:

   L_1 wird zuerst angewandt: $\vec{x}' = L_1\vec{x}$

   L_2 wird als nächstes angewandt: $\vec{x}'' = L_2\vec{x}' = L_2\left(L_1\vec{x}\right)$

   L_3 wird angewandt: $\vec{x}''' = L_3\vec{x}'' = L_3\left(L_2\left(L_1\vec{x}\right)\right)$ usw.

   Am Ende steht: $\vec{x}^{'k} = L_k\left(L_{k-1}\cdots\left(L_1\vec{x}\cdots\right)\right)$

   Das ist das Gleiche wie: $\vec{x}^{'k} = \left(L_k L_{k-1}\cdots L_1\right)\vec{x}$

   Die verkettete Transformationsmatrix L ist folglich gleich: $L = L_k L_{k-1}\cdots L_1$

   Man kann also zusammenfassen:

   Lineare Transformationen werden verkettet, indem ihre Transformationsmatrizen in umgekehrter Reihenfolge miteinander multipliziert werden, oder anders ausgedrückt, der Reihenfolge nach links-seitig verknüpft werden. Die resultierende Transformation bezeichnet man als verkettete Transformation. Will man allgemein zwei lineare Transformationen und ihre Transformationsmatrizen A und B zu einer neuen linearen Transformation mit der Transformationensmatrix C in der Reihenfolge „zuerst A dann B " verketten, so wird dies durch: $C = BA$ erreicht.

2. **Umkehrung/Inversion**

   Die Umkehrung einer Transformation entspricht der Invertierung ihrer Transformationsmatrix.

Allgemein kann festgestellt werden, dass es lineare Transformationen gibt, die nicht umgekehrt werden können, da es Transformationsmatrizen gibt, die nicht invertiert werden können.

Hierzu seien zwei Beispiele genannt:

- $A=\begin{bmatrix} 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{bmatrix}$
- oder allgemein Transformationsmatrizen für deren Determinante gilt: $det(T)=0$

Normalerweise werden von Raytracern ausschließlich Transformationen unterstützt, die auch umgekehrt werden können.

Dass dies keine sonderliche Beschränkung bedeutet, soll im Folgenden verdeutlicht werden.

Dazu betrachten wir noch einmal die Haupteigenschaft von inversen Matrizen:
$A A^{-1}=A^{-1} A=I$

Was heißt das genau? Wird eine Transformation mit ihrer Inversen multipliziert, so ergibt das die Einheitsmatrix, welche einer Transformation ohne Wirkung entspricht (ein Vektor wird auf sich selbst abgebildet). Nach obiger Definition bedeutet aber die Matrixmultiplikation auch eine Verkettung von Transformationen. Wird die Inverse auf die ursprüngliche Transformation angewandt (oder umgekehrt), so heben sich die beiden auf, daraus folgt logischerweise, dass die Inverse die Umkehrung einer Transformation entspricht.

Es soll vorausgesetzt sein, dass sich alle Grundtransformationen umkehren lassen (auf dies wird später noch genauer eingegangen). Diese Annahme ist auch intuitiv, da man sich vorstellen kann, dass eine Drehung, durch eine Gegendrehung in die Gegenrichtung, eine Verschiebung durch eine Verschiebung in die Gegenrichtung und eine Vergrößerung bzw. Verkleinerung durch eine Verkleinerung bzw. Vergrößerung aufgehoben werden kann.

Für den Raytracer sind ja nur diese 3 Grundtransformationen und daraus resultierende Verkettungen interessant, deshalb ist die Eigenschaft $(AB)^{-1}=B^{-1} A^{-1}$ äußerst hilfreich.

Sie sagt aus, dass verkettete Matrizen genau dann umgekehrt werden können, wenn die Matrizen, aus deren Verkettung sie entsteht, umgekehrt werden können. Da wir davon ausgehen, dass sich die Grundtransformationen umkehren lassen, lassen sich somit auch alle anderen für den Raytracer interessanten Transformationen umkehren.

Es ist im Moment unbefriedigend, dass noch nicht bewiesen wurde, dass sich Grundtranformationen invertieren lassen bzw. noch nichts über ihre Darstellung in Matrizenform gesagt wurde.

Dies wird in den nächsten Abschnitten nachgeholt.

3.2.3 Transformationen als Basiswechsel

Wie kann man sich das vorstellen? Nehmen wir an, die Koordinaten eines Objektes seien in einer Basis A gegeben. Durch die Annahme einer neuen Basis verändern sich die Koordinaten (bzw. identische Vektoren haben andere Koordinaten bezüglich der neuen Basis). Dies könnte auch durch eine Transformation geschehen.

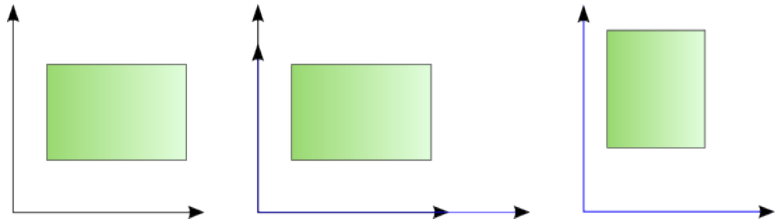

Bild 3.5.Basiswechsel zur Ausführung einer Skalierung (schwarz: alte Basis, blau: neue Basis)

Wie man in Bild 3.6 sieht, kann ein solcher Basiswechsel die Wirkung einer Rotation haben - oder die einer Skalierung, wie in Bild 3.5 zu sehen ist. Die schwarzen Achsen stellen dabei die alten Basisvektoren dar und die blauen Achsen die neuen. Das erste Teilbild stellt das Objekt aus der Sicht (bzw. im Koordinatensystem) der alten Basis dar, das mittlere

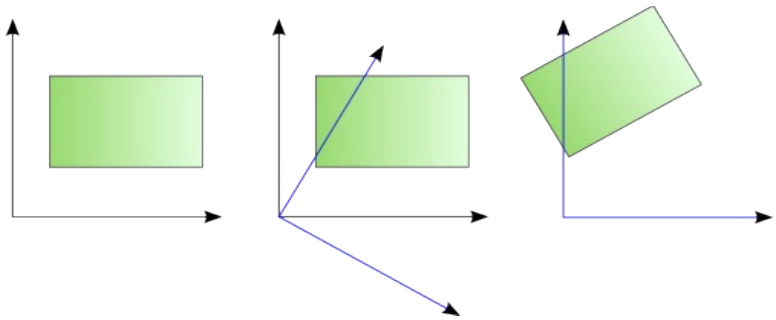

Bild 3.6.Basiswechsel zur Ausführung einer Rotation (schwarz: alte Basis, blau: neue Basis)

veranschaulicht die Lage der neuen Basis im Vergleich zur alten und das rechte zeigt das Objekt aus der Sicht der neuen Basis.

Tatsächlich stellt jede umkehrbare Transformation auch einen Basiswechsel dar bzw. kann durch ihn ausgedrückt werden.

Was genau passiert bei einem Basiswechsel? Ein Vektor wird von einer Basis in eine andere versetzt, in dem seine Koordinaten durch die neuen Basisvektoren ausgedrückt werden.

Mathematisch sieht dies wie folgt aus (im $\mathbb{R}^3$):

$$B=\{\vec{e}_1,\vec{e}_2,\vec{e}_3\}\text{(alte Basis)}$$
$$B'=\{\vec{e}_1',\vec{e}_2',\vec{e}_3'\}\text{(neue Basis)}$$
$$\vec{v}_B=\langle x,y,z\rangle$$
$$\vec{v}_{B'}=\langle x',y',z'\rangle$$
$$x'\vec{e}_1'+y'\vec{e}_2'+z'\vec{e}_3'=x\vec{e}_1+y\vec{e}_2+z\vec{e}_3$$

Seien die Koordinaten der Basisvektoren von B' durch die Basisvektoren von B ausgedrückt, so ergibt sich, falls man auch die Basisvektoren von B durch sich selber ausdrückt[2]:

$$\vec{e}_1=\langle 1,0,0\rangle;\vec{e}_2=\langle 0,1,0\rangle;\vec{e}_3=\langle 0,0,1\rangle$$
$$\vec{e}_1'=\langle e_{1x}',e_{1y}',e_{1z}'\rangle;\vec{e}_2'=\langle e_{2x}',e_{2y}',e_{2z}'\rangle;\vec{e}_3=\langle e_{3x}',e_{3y}',e_{3z}'\rangle$$
$$\Rightarrow x'\begin{pmatrix}e_{1x}'\\e_{1y}'\\e_{1z}'\end{pmatrix}+y'\begin{pmatrix}e_{2x}'\\e_{2y}'\\e_{2z}'\end{pmatrix}+z'\begin{pmatrix}e_{3x}'\\e_{3y}'\\e_{3z}'\end{pmatrix}=\begin{pmatrix}x\\y\\z\end{pmatrix}$$

2 Umgeschrieben zu $x'\vec{e}_1'+y'\vec{e}_2'+z'\vec{e}_3'=\begin{pmatrix}x\\y\\z\end{pmatrix}$ erkennt man, dass dies zum gleichen Ansatz führt, den wir auch im Unterricht benutzen.

Schreibt man dies aus, so erkennt man, dass das Gleiche auch durch Matrizen ausgedrückt werden kann:

$$\begin{bmatrix} e_{1x}' & e_{2x}' & e_{3x}' \\ e_{1y}' & e_{2y}' & e_{3y}' \\ e_{1z}' & e_{2z}' & e_{3z}' \end{bmatrix} \begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

$$\Leftrightarrow \begin{bmatrix} \vec{e}_1' & \vec{e}_2' & \vec{e}_3' \end{bmatrix} \begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \Leftrightarrow$$

$$\Leftrightarrow \begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{bmatrix} \vec{e}_1' & \vec{e}_2' & \vec{e}_3' \end{bmatrix}^{-1} \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

Dazu sei anzumerken, dass eine Matrix, die nur aus Basisvektoren besteht, immer invertierbar ist (für einen Beweis siehe *4.2.1 Matrizen aus Basisvektoren sind invertierbar* im Anhang).

Man kann allgemein festhalten[4]:

> Die Koordinaten $\langle v_1', \cdots, v_n' \rangle$ eines Vektor $\vec{v}$ bezüglich einer Basis B' sind durch die Koordinaten $\langle v_1, \cdots, v_n \rangle$ von $\vec{v}$ bezüglich der Basis B und der Darstellung der Koordinaten der Basisvektoren $\vec{e}_1', \cdots, \vec{e}_n'$ von B' bezüglich B festgelegt mit:
>
> $$\vec{v}_{B'} = \begin{pmatrix} v_1' \\ \vdots \\ v_n' \end{pmatrix} = \begin{bmatrix} \vec{e}_1' & \cdots & \vec{e}_n' \end{bmatrix}^{-1} \begin{pmatrix} v_1 \\ \vdots \\ v_n \end{pmatrix} = \begin{bmatrix} \vec{e}_1' & \cdots & \vec{e}_n' \end{bmatrix}^{-1} \vec{v}_B$$

Dies lässt sich noch weiter vereinfachen, falls vorausgesetzt wird, dass es sich um eine orthonormale Basis handelt[4].

Dann gilt nämlich, dass alle Basisvektoren senkrecht aufeinander stehen und Einheitslänge besitzen.

> Eine Basis ist dann **orthonormal**, wenn ihre Basisvektoren alle orthogonal zueinander sind und die Länge 1 besitzen: $\vec{e}_i \perp \vec{e}_j \Leftrightarrow \vec{e}_i \vec{e}_j = 0$ für $i \neq j$ und $|\vec{e}_k| = 1$ für alle i, j, k

Eine Matrize, deren Zeilen oder Spalten aus den Basisvektoren einer orthonormalen Basis aufgebaut sind, also deren Zeilen- oder Spaltenvektoren senkrecht zueinander stehen und Einheitslänge besitzen, bezeichnet man als **orthonormale Matrix**.

Die Transponierte einer solchen Matrix haben eine angenehme Eigenschaft: sie ist zugleich ihre Inverse!

Der Beweis ist ganz leicht:

O.b.d.A sei die k-te Zeile der Matrix A gleich dem Vektor $\vec{e}_k$ der orthonormalen Basis. Dann bestehen die Spalten der Matrix A^T aus diesen orthonormalen Vektoren.

Wird nun A mit A^T multipliziert, so gilt nach den Regeln der Matrixmultiplikation für die i-te Zeile und j-te Spalte:

$$(A\,A^T)_{ij} = \sum_{k=1}^{n} A_{ik} (A^T)_{kj} = \sum_{k=1}^{n} A_{ik} A_{jk} = \vec{e}_i \vec{e}_j$$

$$\Rightarrow (A\,A^T)_{ij} = \begin{cases} \text{für } i = j : (\vec{e}_i)^2 = |\vec{e}_i|^2 = 1 \\ \text{für } i \neq j : \vec{e}_i \perp \vec{e}_j \Rightarrow \vec{e}_i \vec{e}_j = 0 \end{cases}$$

Und dies entspricht genau der Definition der Einheitsmatrix (Diagonale besteht aus Einsen, der Rest aus Nullen).

Beispiel:

$$A=\begin{bmatrix}0&1&0\\0&0&1\\1&0&0\end{bmatrix}\Rightarrow A^T=\begin{bmatrix}0&0&1\\1&0&0\\0&1&0\end{bmatrix}$$

$$AA^T=\begin{bmatrix}0&1&0\\0&0&1\\1&0&0\end{bmatrix}\begin{bmatrix}0&0&1\\1&0&0\\0&1&0\end{bmatrix}=\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix}=I$$

Zusammengefasst:

> Die Transponierte einer orthonormalen Matrix ist zugleich ihre Inverse.

Wird vorausgesetzt, dass die Basen beim Basiswechsel alle orthonormal sind, so lässt sich diese Aussage auf die Gleichung beim Basiswechsel anwenden:

$$\vec{v}_{B'}=\begin{bmatrix}\vec{e}_1' & \cdots & \vec{e}_n'\end{bmatrix}^{-1}\vec{v}_B=\begin{bmatrix}\vec{e}_1' & \cdots & \vec{e}_n'\end{bmatrix}^{T}\vec{v}_B=\begin{bmatrix}\vec{e}_1'^{\,T}\\ \vdots \\ \vec{e}_n'^{\,T}\end{bmatrix}\vec{v}_B=\begin{pmatrix}\vec{e}_1'\vec{v}_B\\ \vdots \\ \vec{e}_n'\vec{v}_B\end{pmatrix}$$

$\begin{bmatrix}\vec{e}_1'^{\,T}\\ \vdots \\ \vec{e}_n'^{\,T}\end{bmatrix}$ kann dabei als Transformationsmatrix interpretiert werden.

Insgesamt kann man also feststellen, dass Transformationen (in orthogonalen Koordinatensystemen) und Basiswechsel äquivalent sind.

Rotationen lassen sich dadurch überaus leicht konstruieren, da man nur die neuen Basisvektoren aufstellen und in die Transformationsmatrix einsetzen muss.

Beispiel:

Es soll eine Rotation um α Grad im mathematischen Drehsinn (gegen den Uhrzeiger) an der x-y-Ebene (siehe Bild 3.7) durchgeführt werden. Äquivalent dazu ist ein Basiswechsel mit einer Drehung im Uhrzeigersinn.

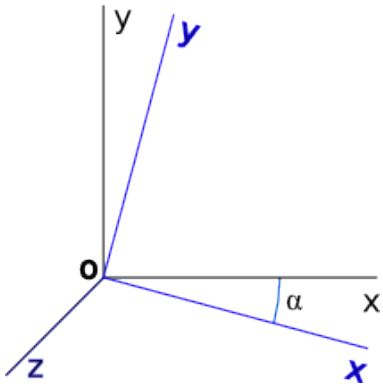

Bild 3.7.Rotation um die Z-Achse (x-y-Ebene)

Als erstes fällt auf, dass die z-Achse unverändert bleibt, da die Drehung ja um sie erfolgt.

D.h. $\vec{e}_3'=\langle 0,0,1\rangle$.

Betrachtet man zusätzlich Bild 3.8, so erkennt man, dass für die neue x-Achse, also $\vec{e}_1'$, die positive x-Achse die Ankathete und die negative y-Achse Gegenkathete darstellen, damit folgt für den neuen Basisvektor: $\vec{e}_1'=\langle \cos\alpha,-\sin\alpha,0\rangle$ Analog folgt für den Basisvektor der neuen y-Achse, für den die positive y-Achse die Ankathete und die positive x-Achse die Gegenkathete darstellen: $\vec{e}_2'=\langle \sin\alpha,\cos\alpha,0\rangle$

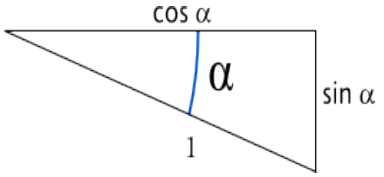

Bild 3.8.Rechtwinkliges Dreieck

Diese 3 Basisvektoren haben alle Einheitslänge und stehen senkrecht aufeinander, wie sich leicht durch ein paar Punktprodukte oder logisches Schlussfolgern überprüfen lässt.

Damit stellen sie eine neue orthonormale Basis dar und eine Rotation um die z-Achse um α Grad (gegen den Uhrzeigersinn) kann ausgedrückt werden durch die Transformationsmatrix[3]:

$$T=\begin{bmatrix}\cos\alpha & -\sin\alpha & 0\\ \sin\alpha & \cos\alpha & 0\\ 0 & 0 & 1\end{bmatrix}$$

Die Rotationen um die x- und y-Achse lassen sich analog herleiten.

Dass Rotationsmatrizen invertierbar sind, wurde oben gezeigt, da jede beliebige Rotation die Orthonormalität der Basis erhält und somit jede Rotation einen Basiswechsel darstellt.

Damit wurde auch mathematisch bewiesen, dass eine der 3 verwendeten Grundtransformationen invertierbar ist.

3.2.4 Darstellung von Skalierungen durch Matrizen

Jede Skalierung entlang einer beliebigen Achse kann auf eine Streckung entlang der 3 Koordinatenachsen durch Rotation zurückgeführt werden, indem das Objekt so rotiert wird, dass die Achse mit einer Koordinatenachse zusammenfällt (siehe Bild 3.9).

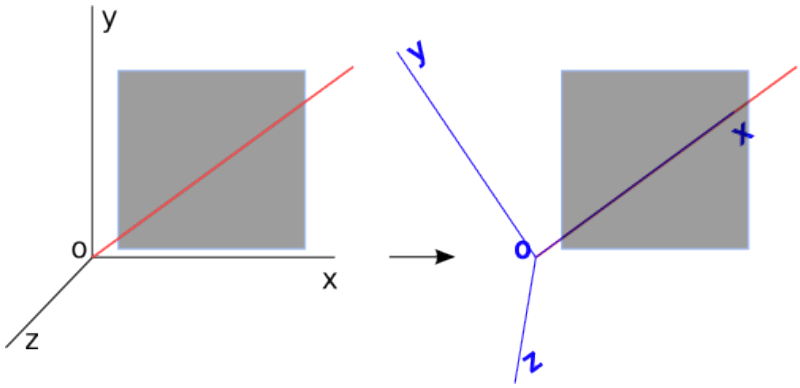

Bild 3.9.Die Koordiantenachsen (schwarz) werden so rotiert, dass dann eine davon (blau) mit der Skalierungsachse (rot) zusammenfällt

Deshalb reicht es aus, nur eine Skalierung entlang der Koordinatenachsen zu betrachten. Die Komponentenfunktionen einer solchen Transformation S, die die x-, y-, und z-Koordinaten um s_x, s_y, und s_z streckt, sehen wie folgt aus:

$$S_1(\vec{x})=s_x\cdot x_1$$
$$S_2(\vec{x})=s_y\cdot x_2$$
$$S_3(\vec{x})=s_z\cdot x_3$$

Die dazugehörige Transformationsmatrix ist also, wie man leicht erkennt:

$$S=\begin{bmatrix}s_x & 0 & 0\\ 0 & s_y & 0\\ 0 & 0 & s_z\end{bmatrix}$$

Da diese Matrix zwar orthogonal aber nicht orthonormal ist, lässt sich die Inverse nicht einfach durch Transponieren gewinnen. Ein intuitiver Ansatz hilft hier:

Wen man einen Vektor um s_x, s_y, bzw. s_z streckt, so wird dies wohl aufgehoben werden können, wenn man ihn um $1/s_x$, $1/s_y$, bzw. $1/s_z$ staucht.

Durch Probe erhält man tatsächlich:

$$\begin{bmatrix}s_x & 0 & 0\\ 0 & s_y & 0\\ 0 & 0 & s_z\end{bmatrix}\cdot\begin{bmatrix}1/s_x & 0 & 0\\ 0 & 1/s_y & 0\\ 0 & 0 & 1/s_z\end{bmatrix}=\begin{bmatrix}s_x/s_x & 0 & 0\\ 0 & s_y/s_y & 0\\ 0 & 0 & s_z/s_z\end{bmatrix}=\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix}$$

3 Ein anderer Ansatz zur Herleitung kann in [4] gefunden werden.

Damit ist auch die zweite von 3 Grundtransformationen als Transformationsmatrix beschrieben worden und es wurde gezeigt, wie sie invertiert werden kann.

3.2.5 Translationen und affine Transformationen

Eine Translation/Verschiebung eines Vektors $\vec{v}=\langle v_1, v_2, v_3\rangle$ um den Verschiebungsvektor $\vec{d}$ kann durch folgende Transformation beschrieben werden:

$$v'=\vec{T}(\vec{v})=\vec{v}+\vec{d}$$

Eine solche Transformation wird übrigens auch als **affine Transformation** bezeichnet[9]:

> Eine Transformation, der Art $\vec{v}'=\vec{T}(\vec{v})=A\vec{v}+\vec{d}$, wobei A eine lineare Transformationsmatrix ist, die jeden Vektor $\vec{v}$ nicht nur linear transfomiert, sondern zusätzlich auch um einen konstanten **Verschiebungsvektor** $\vec{d}$ verschiebt, heißt **affine Transformation**.

Ein Problem ergibt sich daraus, dass $\vec{d}$ eine Konstante ist und lineare Transformationen nur aus ersten Potenzen der Vektorkomponenten bestehen und somit keine Konstanten in linearen Transformationen vorkommen können.

Deswegen kann T nicht als lineare Transformation im $\mathbb{R}^3$ Vektorraum ausgedrückt werden und besitzt auch keine 3x3 Transformationsmatrix.

Aber dieses Problem kann durch einen Trick gelöst werden:

Stellen wir dazu die Funktionsvorschrift von T auf:

$$\vec{T}(\vec{v})=\begin{pmatrix} T_1(\vec{v})=1\cdot v_1+d_1\cdot 1 &=& 1\cdot v_1+0\cdot v_2+0\cdot v_3+d_1\cdot 1 \\ T_2(\vec{v})=1\cdot v_2+d_2\cdot 1 &=& 0\cdot v_1+1\cdot v_2+0\cdot v_3+d_1\cdot 1 \\ T_3(\vec{v})=1\cdot v_3+d_3\cdot 1 &=& 0\cdot v_1+0\cdot v_2+1\cdot v_3+d_1\cdot 1 \end{pmatrix}$$

Die Art und Weise der 2. Umformung sieht so aus, als ob sie durch Matrizendarstellung vereinfacht werden könnte und tatsächlich:

$$\vec{T}(\vec{v})=\begin{bmatrix} 1 & 0 & 0 & d_1 \\ 0 & 1 & 0 & d_2 \\ 0 & 0 & 1 & d_3 \end{bmatrix}\begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ 1 \end{bmatrix}=\begin{bmatrix} 1 & 0 & 0 & d_1 \\ 0 & 1 & 0 & d_2 \\ 0 & 0 & 1 & d_3 \end{bmatrix}\begin{bmatrix} \vec{v} \\ 1 \end{bmatrix}$$

Dies sieht schon fast wie eine Transformationsmatrix aus, wobei der Vektor des $\mathbb{R}^3$ in einen Vektor des $\mathbb{R}^4$ umgewandelt wurde, indem eine 1 als 4. Komponente hinzugefügt wurde. Die Matrix muss nun noch quadratisch gemacht werden, damit solche Transformationen verkettet werden können[4]. Wie könnte also die 4. Zeile aussehen? Die 1 scheint essentiell zu sein, damit der Verschiebungsvektor erhalten bleibt, d.h. das Punktprodukt des vierten Zeilenvektors/der vierten Zeile der Matrix mit $\langle v_1, v_2, v_3, 1\rangle$ sollte auch wieder 1 ergeben. Dies ist nur dann immer möglich, wenn die 4. Zeile aus $\langle 0, 0, 0, 1\rangle$ besteht.

Die Transformationsmatrix sieht damit wie folgt aus:

$$T=\begin{bmatrix} 1 & 0 & 0 & d_1 \\ 0 & 1 & 0 & d_2 \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

4 Man erinnere sich: eine 3x4 Matrix multipliziert mit einer 4x1 Matrix ergibt eine 3x1 Matrix, die nicht weiter mit einer 3x4 Matrix multipliziert werden kann.

Dabei ist T selbst eine lineare Transformationsmatrix! Dies steht aber nicht im Widerspruch zu dem oben genannten Problem. T ist eine lineare Transformationsmatrix für Vektoren des $\mathbb{R}^4$, aber nicht für Vektoren des $\mathbb{R}^3$! Für diese handelt es sich um eine affine Transformation. Erst durch die Umwandlung in Vektoren des $\mathbb{R}^4$ kann die affine Transformation der $\mathbb{R}^3$ -Vektoren durch eine lineare Transformation dargestellt werden.

Diese Hilfsvektoren des $\mathbb{R}^4$ Vektorraumes bezeichnet man als **homogene Vektoren**, bzw. ihre Koordinaten als **homogene Koordinaten**[5][8].

Wie oben schon erwähnt, werden „normale" $\mathbb{R}^3$ Vektoren in homogene Vektoren umgewandelt, indem die neue, 4. Komponente gleich 1 gesetzt wird.

Die Achse, entlang der die 4. Komponente angetragen wird, bezeichnet man geläufig als **w-Achse**.

Es ist aber auch möglich sie gleich 0 zu setzen. Die Folge davon ist, dass der Vektor dann nicht verschoben wird, da in diesem Falle die Komponenten des Verschiebungsvektors mit 0 multipliziert werden:

$$\begin{bmatrix} 1 & 0 & 0 & d_1 \\ 0 & 1 & 0 & d_2 \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ 0 \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ 0 \end{bmatrix}$$

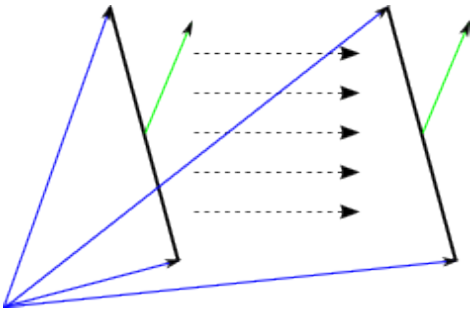

Bild 3.10.Richtiger Effekt einer Translation auf Richtungs- (grün) und Ortsvektoren (blau)

Diese Umwandlung ist für Richtungsvektoren nützlich, da diese von Verschiebungen auch nicht betroffen werden sollen, während aber Ortsvektoren verschoben werden sollen (siehe Bild 3.10).

Homogene Koordinaten werden in normale Vektoren umgewandelt, indem man einfach wieder die 4. Komponente weglässt.

Zusammengefasst kann man also schreiben:

Ein **homogener Vektor** ist ein Vektor des $\mathbb{R}^4$ Vektorraumes, der es ermöglicht, **affine Transformationen** von $\mathbb{R}^3$ Vektoren als **lineare Transformationen** im $\mathbb{R}^4$ Raum zu betrachten.

Ein **Ortsvektor** $\vec{v}$ wird zu einem **homogenen Vektor** erweitert, indem seine 4. Komponente 1 gesetzt wird:

$$\vec{v}_{homogen} = \begin{pmatrix} \vec{v} \\ 1 \end{pmatrix}$$

Ein **Richtungsvektor** $\vec{r}$ wird zu einem **homogenen Vektor** erweitert, indem seine 4. Komponente 0 gesetzt wird:

$$\vec{r}_{\text{homogen}} = \begin{pmatrix} \vec{r} \\ 0 \end{pmatrix}$$

Ein **homogener Vektor** kann in einen **Orts-** oder **Richtungsvektor** umgewandelt werden, in dem die 4. Komponente (entweder 1 oder 0) weggelassen wird:

$$\vec{x}_{\text{homogen}} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} \Rightarrow \vec{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$$

Man kann Translationen nur dann durch lineare Transformationen darstellen, wenn man homogene Koordinaten und quadratische 4x4 Matrizen benutzt.

> 4x4 Matrizen, die verwendet werden, um homogene Vektoren zu transformieren, werden als **homogene Matrizen** bezeichnet.

Der Wechsel auf 4x4 Matrizen fällt nicht schwer, da alle Definitionen und (skizzierten) Beweise unabhängig vom Rang der Matrix oder des Vektorraumes waren.

Allgemein kann eine 3x3 Transformationsmatrix A leicht in eine homogene Transformationsmatrix H umgewandelt werden durch:

$$H = \begin{bmatrix} & & & 0 \\ & A & & 0 \\ & & & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Eine solche homogene Transformationsmatrix beschreibt eine lineare, homogene Transformation.

Die Eigenschaft der Verkettung folgt analog aus den Eigenschaften der linearen Transformation.

Bei der Invertierbarkeit kann man bei den Transformationsmatrizen der Grundtransformationen der Rotation und Skalierung wie schon beschrieben argumentieren und damit ihre Invertierbarkeit beweisen.

Die Umkehrung der Translation kann intuitiv wie die Umkehrung der Skalierung hergeleitet werden:

Wenn man alles um einen Verschiebungsvektor $\vec{d}$ verschiebt, so kann dies durch eine Verschiebung um $-\vec{d}$ rückgängig gemacht werden.

Die Probe zeigt, dass diese Annahme korrekt ist:

$$\begin{bmatrix} 1 & 0 & 0 & d_1 \\ 0 & 1 & 0 & d_2 \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} 1 & 0 & 0 & -d_1 \\ 0 & 1 & 0 & -d_2 \\ 0 & 0 & 1 & -d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & -d_1+d_1 \\ 0 & 1 & 0 & -d_2+d_2 \\ 0 & 0 & 1 & -d_3+d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = I$$

Allgemein kann man über Translationen und affine Transformation noch festhalten:

> Eine **affine Transformation** der Form $\vec{v}' = \vec{T}(\vec{v}) = A\vec{v} + \vec{d}$, wobei **Translationen** ein Spezialfall mit $A = I$ sind, kann ausgedrückt werden durch eine **homogene Transformationsmatrix** der Form:
>
> $$T = \begin{bmatrix} & A & & \vec{d} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Durch die gleiche Argumentation wie in *3.2.2 Transformationen und Matrizen* kann jetzt auch wieder bewiesen werden, dass verkettete Transformationen aus diesen Grundtransformationen invertiertbar sind.

Damit ist der Wechsel in den homegenen Koordinatenraum (also die Verwendung von homogenen Koordinaten und Transformationen/Matrizen) schon fast vollständig behandelt.

3.2.6 Umkehrung von homegenen Transformationsmatrizen

Es fehlt noch ein Hilfsmittel um eine homogene 4x4 Transformationsmatrizen T der Form $T=\begin{bmatrix} & A & & \vec{d} \\ 0 & 0 & 0 & 1 \end{bmatrix}$ zu invertieren, wobei A eine 3x3 Transformationsmatrix ist.
Es kann leicht gezeigt werden, dass der 4. Zeilenvektor von 2 solchen miteinander verketteten Matrizen auch $\langle 0,0,0,1 \rangle$ ist. Dies kann durch Ausmultiplizieren leicht überprüft werden.

Um den Aufbau der Inversen herzuleiten, analysieren wir zuerst einmal das Verhalten der ursprünglichen Matrix bei einer Transformation:

$$\vec{v}\,' = \begin{pmatrix} v_1\,' \\ v_2\,' \\ v_3\,' \\ v_4\,' \end{pmatrix} = T\,\vec{v} = \begin{pmatrix} A\begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix} + \vec{d}\cdot v_4 \\ v_4 \end{pmatrix}$$

Versucht man nach $\vec{v}$ aufzulösen, so erhält man nacheinander, wenn man die 4. Komponente aus den Betrachtungen weglässt, da offensichtlich gilt $v_4 = v_4\,'$:

$$\begin{pmatrix} v_1\,' \\ v_2\,' \\ v_3 \end{pmatrix} = A\begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix} + \vec{d}\cdot v_4 \Leftrightarrow \begin{pmatrix} v_1\,' \\ v_2\,' \\ v_3\,' \end{pmatrix} - \vec{d}\cdot v_4 = A\begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix} \Leftrightarrow$$

$$\overset{\text{mit } v_4 = v_4\,'}{\Leftrightarrow} \begin{pmatrix} v_1\,' \\ v_2\,' \\ v_3\,' \end{pmatrix} - \vec{d}\cdot v_4\,' = A\begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix} \Leftrightarrow A^{-1}\left(\begin{pmatrix} v_1\,' \\ v_2\,' \\ v_3\,' \end{pmatrix} - \vec{d}\cdot v_4\,'\right) = A^{-1}A\begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix} = \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix}$$

$$\Leftrightarrow A^{-1}\begin{pmatrix} v_1\,' \\ v_2\,' \\ v_3\,' \end{pmatrix} - A^{-1}\vec{d}\cdot v_4\,' = \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix} \Leftrightarrow \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix} = A^{-1}\begin{pmatrix} v_1\,' \\ v_2\,' \\ v_3\,' \end{pmatrix} - \left(A^{-1}\vec{d}\right)\cdot v_4\,'$$

Fügt man die 4. Komponente wieder hinzu, so ergibt sich:

$$\begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{pmatrix} = \begin{pmatrix} A^{-1}\begin{pmatrix} v_1\,' \\ v_2\,' \\ v_3\,' \end{pmatrix} - \left(A^{-1}\vec{d}\right)\cdot v_4\,' \\ v_4\,' \end{pmatrix}$$

Vergleicht man dies mit dem Ansatz für die Transformation von $\vec{v}$ zu $\vec{v}\,'$, so erkennt man, dass es sich bei der Umkehrung um eine Transformation gleichen Typs handeln muss. Schreibt man diese analog in eine homogene Transforamtionsmatrix um, so erhält man:

$$T^{-1} = \begin{bmatrix} & A^{-1} & & -A^{-1}\vec{d} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Eine Probe zeigt, dass die hergeleitete Inverse korrekt ist:

$$T\,T^{-1} = \begin{bmatrix} & A & & \vec{d} \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} & A^{-1} & & -A^{-1}\vec{d} \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} & A\,A^{-1} & & -A\,A^{-1}\vec{d} + \vec{d} \\ 0 & 0 & 0 & 1 \end{bmatrix} =$$

$$= \begin{bmatrix} & I & & -\vec{d} + \vec{d} \\ 0 & 0 & 0 & 1 \end{bmatrix} = I$$

Man kann also festhalten:

Ist T die Transformationsmatrix einer homogenen Transformation mit:

$$T=\begin{bmatrix} & A & & \vec{d} \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

wobei A eine 3x3 Transformationsmatrix ist, so ist die Inverse dieser Matrix gegeben durch:

$$T^{-1}=\begin{bmatrix} & A^{-1} & & -A^{-1}\vec{d} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

3.2.7 Transformation von (Ebenen)normalen

Als letztes soll noch untersucht werden, wie Ebenennormalen (im Gegensatz zu Orts- und Richtungsvektoren) transformiert werden müssen.

Die Eigenschaft einer Normale ist, dass sie zu einer Ebene immer senkrecht steht. Liegt mit $\vec{a}$ ein Richtungsvektor in einer Ebene, so muss für die dazugehörige Ebenennormale $\vec{n}$ immer gelten: $\vec{a}\,\vec{n}=0$.
Diese Bedingung kann auch in Matrizenschreibweise ausgedrückt werden durch: $\vec{a}^T\vec{n}=0$

Wenn $\vec{a}$ nun durch eine Transformation mit der Transformationsmatrix T transformiert wird, so muss $\vec{n}$ auch transformiert werden. Nennen wir diese noch unbekannte Transformationsmatrix X.

Es muss also dann immer noch gelten:
$(T\vec{a})^T(X\vec{n})=0\Leftrightarrow\vec{a}^T T^T X\vec{n}=0$

Wie muss X beschaffen sein, damit T^T herausfällt? X muss die Inverse von T^T sein!
Setzt man $X=\left(T^T\right)^{-1}$ in die Gleichung ein, so erhält man:

$$\vec{a}^T T^T X\vec{n}=0\Leftrightarrow\vec{a}^T T^T\left(T^T\right)^{-1}\vec{n}=\vec{a}^T I\vec{n}\Leftrightarrow\vec{a}^T\vec{n}=0$$

Daraus folgt, dass man eine Normale mit der Inversen der transponierten Transformationsmatrix transformieren muss, um ihre Eigenschaft als Ebenennormale zu erhalten!

Für orthonormale Transformationsmatrizen ist gerade die Transponierte die Inverse. Folglich gilt für solche Transformationen, dass Normalen mit der gleichen Transformationsmatrix transformiert werden können wie andere Vektoren auch. Bei nicht symmetrischen Skalierungen trifft dies aber nicht zu (siehe Bild 3.11).

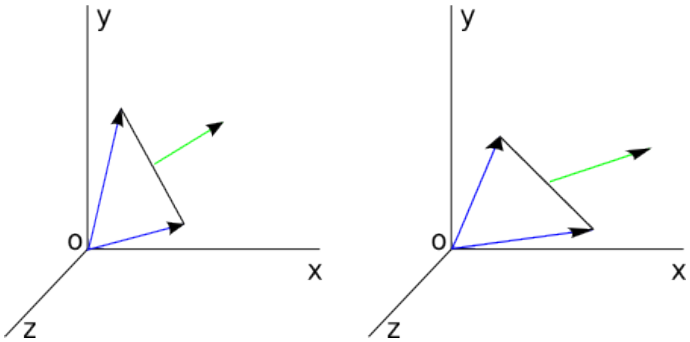

Bild 3.11.Ungleichmäßige Skalierung (blau: Ortsvektoren, grün: rechts mit gleicher Transformationsmatrix inkorrekt transformierte Normale)

3.3 Strahlenverfolgung und Kontaktpunktbestimmung

Im Überblick in *2.3 Rendervorgang* wurde schon erklärt, dass für die Strahlen die Kontaktpunkte mit Objekten der Szene bestimmt werden und dabei die Strahlen in den Koordinatenraum des jeweiligen Objektes versetzt werden, um die Anzahl der durchzuführenden Transformationen pro Objekt zu minimieren.

Es soll noch eine mathematische Definition eines Strahles aufgeführt werden:

Ein **Strahl** (engl. Ray) R ist eine Halbgerade, die durch einen Startpunkt oder Ursprung S, bzw. dessen Ortsvektor $\vec{s}$ und einem Richtungsvektor $\vec{r}$ eindeutig festgelegt ist:
$R:t\to\vec{R}(t)=\vec{s}+t\vec{r}$ bzw. $R:\vec{x}=\vec{s}+t\vec{r}$ und $t\in\mathbb{R}_0^+$

Mit dieser Darstellung lassen sich mathematisch alle Kontaktpunkte mit verschiedenen Objekten bestimmen. Außerdem lässt sich mit einfachen Vergleichen durch den Zusammenhang zwischen dem Parameter und der Entfernung vom Startpunkt des Strahles feststellen mit welchem Objekt der Strahl zuerst Kontakt hat: Ein größeres t bedingt eine größere Entfernung vom Startpunkt, ein kleineres eine kleinere.

3.3.1 Kontakt mit einer Ebene

Um den Kontaktpunkt eines Strahles mit einer Ebene zu bestimmen, muss zuerst festgelegt werden, wie Ebenen beschrieben werden sollen[7]:

> **Ebenen** werden durch die Hessesche Normalenform der Ebenengleichung dargestellt: $\vec{p} \in E \Leftrightarrow \vec{n}\,\vec{p} = d$, wobei $\vec{n}$ die normalisierte Ebenennormale und d der entlang der Ebenennormale gerichtete Abstand der Ebene zum Ursprung ist.

Um den Kontaktpunkt K (bzw. seinen Ortsvektor $\vec{k}$) eines Strahles R mit einer Ebene $E(\vec{n}, d)$ zu bestimmen, setzt man die Strahlenfunktion in die Ebenengleichung ein:

$$\vec{n} \cdot \vec{R}(t) = d \Leftrightarrow \vec{n} \cdot (\vec{s} + t\,\vec{r}) = d \Leftrightarrow \vec{n}\,\vec{s} + t\,\vec{n}\,\vec{r} = d \Leftrightarrow t\,\vec{n}\,\vec{r} = d - \vec{n}\,\vec{s}$$

$$\Leftrightarrow t = \frac{d - \vec{n}\,\vec{s}}{\vec{n}\,\vec{r}} \text{ für } \vec{n} \cdot \vec{d} \neq 0$$

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, d.h. ob der Kontaktpunkt sich auch wirklich auf dem Strahl befindet. K bzw. $\vec{k}$ lässt sich dann durch Einsetzen von t 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.

Dazu 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 mit dem Ortsvektor $\vec{p}$ befindet sich im nicht-positiven Halbraum einer Ebene $E(\vec{n}, d)$, wenn gilt: $\vec{n}\,\vec{p} \leq d$

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 größer als die bisher „beste" ist, wird abgebrochen und zu Schritt 6 gesprungen
4. Überprüfe, ob der Punkt Teil der positiven Halbräume einer der anderen Ebenen ist. Falls zutreffend, liegt der Punkt außerhalb des Polyeders, dann wird abgebrochen und zu Schritt 6 gesprungen
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.

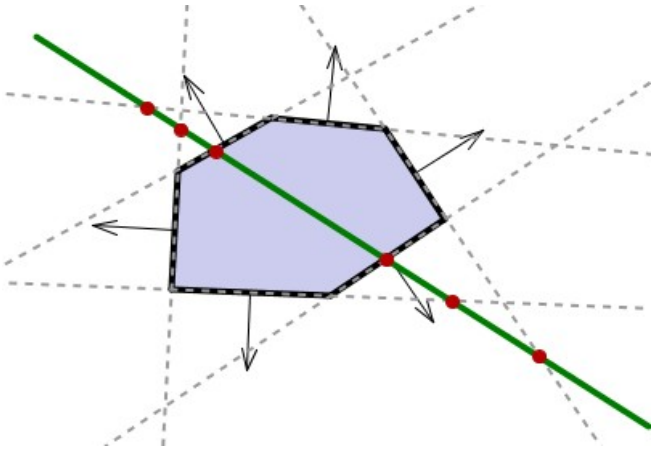

Bild 3.12.Beispiel an einem unregelmäßigen 6-Eck

Der Algorithmus soll an einem Beispiel (Bild 3.12) veranschaulicht werden, aber da dieses Beispiel im zwei Dimensionalen stattfindet, werden durch die hessesche Normalenform keine Ebenen sondern Geraden beschrieben:

Die grau gestrichelten Linien stellen die Geraden dar, die durch hessesche Normalenform im zwei Dimensionalen beschrieben werden. Die Vektoren sind die Normalenvektoren, dieser Geraden bzw. der Seiten des 6-Eckes. Die blaue Fläche stellt die Schnittmenge der negativen Halbräume aller durch die Normalenform beschriebenen Geraden dar. Man erkennt, dass diese der Fläche des 6-Eckes entspricht. Die grüne Halbgerade stellt den Strahl dar und die roten Punkte die Schnittpunkte zwischen den Geraden und dem Strahl.

Der Algorithmus geht nun wie folgt vor: er bestimmt alle Schnittpunkte der Halbgerade mit den Geraden: 6 Schnittpunkte für 6 Geraden.

Der Algorithmus muss für jeden der Schnittpunkte entscheiden, ob dieser auch wirklich ein Kontaktpunkt ist. Er überprüft dies, indem er prüft, ob der Schnittpunkt Teil der Schnittfläche der nicht-positiven Halbräume ist. Ist dies nicht der Fall, so liegt der Punkt außerhalb. Von den wirklichen Kontaktpunkten wird dann jener ausgewählt, der am nächsten zum Startpunkt des Strahles liegt.

3.3.3 Kontakt mit einer Kugel(oberfläche)

Eine Kugeloberfläche 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, $\vec{p}$ der Punkt auf der Kugeloberfläche und *r* ihr Radius ist.
Der **erste** Kontaktpunkt lässt sich dann wieder durch Einsetzen finden:

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

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

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

$$D=b^2-4\mathrm{ac}=4\left(\vec{r}\cdot\vec{o}\right)^2-4\,\vec{r}^2\left(\vec{o}^2-r^2\right)$$

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

Für $D=0$ gibt es einen, für $D>0$ zwei mögliche Kontaktpunkte, ansonsten keinen.

Es interessieren nur nicht-negative Lösungen, da diese auf dem Strahl liegen, außerdem interessiert bei 2 nicht-negativen Lösungen, die kleinere, da diese eine kleinere Kontaktdistanz aufweist und somit den eigentlichen Kontakt beschreibt.

Der Kontaktpunkt bzw. sein Ortsvektor lässt sich wieder durch einsetzen in $\vec{R}(t)$ berechnen.

3.4 Farbberechnung und Schattierungsmodell

Bei dem Schattierungs-/Beleuchtungsmodell handelt ist sich meistens um eine grobe Vereinfachung der physikalischen Gegebenheiten. Moderne Raytracer benutzen meistens ausgeklügelte Modelle, doch ich möchte mich auf ein sehr einfaches Modell stützen, da alles andere zu sehr in die Physik geht[4].

Es wurde schon erwähnt, dass Farben durch 3 Einzelwerte gespeichert werden, die die verschiedenen Intensitäten der 3 Grundfarben, Rot, Grün und Blau, enthalten. Mit Hilfe dieser Werte kann die Farbe durch additive Farbmischung dann bestimmt werden.

Sei C die Farbe des von einem Kontaktpunkt in Richtung das Betrachters/Strahlursprungs gesendeten Lichtes, wobei $C \in \mathbb{R}^3$, aber kein Vektor ist, sondern einfach ein 3-er Tupel, das die 3 Intensitäten enthält.

Um C zu berechnen, muss man verschiedene Einzeleinflüsse auf C auswerten:

- Szenen-spezifische Lichteinflüsse: sogenanntes **ambientes** Licht, „Umgebungslicht", dass von überall her einstrahlt und die Szene gleichmäßig ausleuchtet
- Lichtquellen-spezifsche Einflüsse, die für jede Lichtquelle einzeln betrachtet werden müssen: der sogenannte **diffuse** Anteil, der durch Streuung des Lichtes an der Kontaktoberfläche entsteht und der sogenannte **spekulare** Anteil, der durch direkt reflektiertes Licht entsteht.
- Material-spezifische Einflüsse, die sich aus den Materialeigenschaften ergeben

Um das ganze durch eine Formel ausdrücken zu können, müssen zuerst einige Bezeichnungen festgelegt werden.

$M_{ambient}$, M_{diffus} und $M_{spekular}$, mit Werten zwischen 0 und 1 für jede der 3 Farbkomponenten, bezeichnen die Stärke, mit der das Material des Kontaktobjektes auf die ambienten, diffusen und spekularen Einflüsse reagiert.
$S_{ambient}$ bezeichnet die Intensitäten des ambienten Szenenlichtes.
L_{diffus} und $L_{spekular}$ stellen die diffusen und spekularen Lichtintensitäten einer Lichtquelle L dar und $\vec{s}_L$ den Ursprung des Lichtes, bzw. die Position der Lichtquelle.
$I_{diffus}(\vec{k}, \vec{n}, \vec{s}_L)$ ist eine Funktion, die in Abhängigkeit vom Ortsvektor $\vec{k}$ des Kontakts, der Oberflächennormale $\vec{n}$ am Kontaktpunkt und dem Ortsvektor $\vec{s}_L$ der Position einer Lichtquelle L die Intensität berechnet, mit welcher der diffuse Anteil des Lichtes den Betrachter erreichen kann.
$I_{spekular}(\vec{k}, \vec{n}, \vec{s}_L, \vec{v})$ bezeichnet analog die Funktion, die dies für den spekularen Anteil berechnet, aber zusätzlich noch den Ortsvektor $\vec{v}$ der Betrachterposition braucht.

Farbtupel werden komponentenweise addiert und multipliziert, also:

$$(r_1, g_1, b_1) + (r_2, g_2, b_2) = (r_1 + r_2, g_1 + g_2, b_1 + b_2)$$
$$(r_1, g_1, b_1) \cdot (r_2, g_2, b_2) = (r_1 \cdot r_2, g_1 \cdot g_2, b_1 \cdot b_2)$$

Mit diesem Wissen kann man die Formel für C wie folgt ausdrücken:

$$C = M_{ambient} \cdot S_{ambient} + \sum_{L = \text{sichtbares Licht}} \Big[M_{diffus} \cdot L_{diffus} \cdot I_{diffus}\left(\vec{k}, \vec{n}, \vec{s}_L\right) + M_{spekular} \cdot L_{spekular} \cdot I_{spekular}\left(\vec{k}, \vec{n}, \vec{s}_L\right) \Big]$$

Wie werden $I_{diffus}(\vec{k}, \vec{n}, \vec{s}_L)$ und $I_{spekular}(\vec{k}, \vec{n}, \vec{s}_L)$ berechnet?

Diffuser Anteil

Der diffuse Anteil, der gestreutes Licht entsteht, ist unabhängig vom Beobachterstandpunkt, da angenommen wird, dass der diffuse Anteil überall hin gleichmäßig gestreut wird.

Einzig und allein der Winkel zwischen der Kontaktoberfläche und dem Lichteinfallsvektor verändert die Streuintensität.

Dies lässt sich wie folgt begründen (Idee aus[4]):

Nehme man an, ein Lichtstrahl würde senkrecht auf eine Oberfläche fallen und die Querschnittsfläche wäre A, neigt man die Fläche gegenüber dem Lichtstrahl, so vergrößert sich A um $1/\cos\alpha$, wie man Bild 3.13 entnehmen kann. Die transportiere Lichtleistung ist aber konstant geblieben, da sich der Durchmesser des Lichtstrahles nicht verändert hat. Daraus kann man schließen, dass sich die Intensität der pro Flächeneinheit auftreffenden Lichtleistung um $\cos\alpha$ verringert hat.

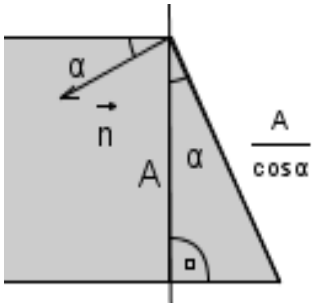

Bild 3.13.Lichteinfall dargestellt als grauer Balken

Außerdem erkennt man, dass sich der Neigungswinkel auch allgemein zwischen dem Einfallsvektor des Lichtes und der geneigten Oberflächenormale (im Bild $\vec{n}$) wiederfindet.

Damit lässt sich die Funktion $I_{diffus}(\vec{k},\vec{n},\vec{s}_L)$ aufstellen:

$$I_{diffus}(\vec{k},\vec{n},\vec{s}_L)=max(0,\cos\alpha)=max\left(0,\frac{\vec{n}\cdot\overbrace{(\vec{s}_L-\vec{k})}^{\text{Lichteinfallsvektor}}}{|\vec{n}||\vec{s}_L-\vec{k}|}\right)$$

Dabei wird $max(0,\cdots)$ benutzt, um den Wert nicht negativ werden zu lassen, da die Intensität nur zwischen 0 und 1 liegen soll.

Spekularer Anteil

Der spekulare Anteil entsteht dadurch, dass Licht nach dem Reflexionsgesetz „Einfallswinkel=Ausfallswinkel" reflektiert wird und den Beobachter trifft.

Wenn der Vektor vom Kontaktpunkt in Richtung Betrachter genau mit dem reflektierten Einfallsvektor des Lichtes übereinstimmt, sollte der ganze spekulare Anteil übertragen werden, ansonsten soll er mit größer werdender Diskrepanz zwischen den beiden Vektoren immer kleiner werden.

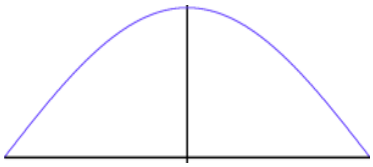
Bild 3.14.Graph des Cosinus von $-\pi/2$ bis $+\pi/2$

Auch hier erweist sich das Punktprodukt bzw. der Cosinus zwischen den beiden Vektoren als nützlich, da er bei 0° sein Maximum hat und sonst abnimmt (vgl. Bild 3.14).

Bevor aber die Funktion angegeben wird, erscheint es ratsam zuerst eine Hilfsfunktion $\vec{r}(\vec{v},\vec{n})$ herzuleiten, die einen Vektor $\vec{v}$ an einer Normalen $\vec{n}$ reflektiert.

Dazu braucht man einen Hilfsvektor $\vec{p}$, der eine der Katheten in dem aus $\vec{n}$, $\vec{v}$ und $\vec{p}$ gebildeten rechtwinkligen Dreieck ist:

$\vec{p}=\frac{\vec{v}\cdot\vec{n}}{|\vec{n}|}\cdot\frac{\vec{n}}{|\vec{n}|}-\vec{v}$ (Herleitung, siehe [4])

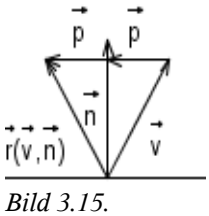

Bild 3.15.

Der reflektierte Vektor ist dann gegeben durch:

$$\vec{r}(\vec{v},\vec{n})=\vec{v}+2\,\vec{p}=\vec{v}+2\left(\frac{\vec{v}\cdot\vec{n}}{|\vec{n}|}\cdot\frac{\vec{n}}{|\vec{n}|}-\vec{v}\right)=2\frac{\vec{v}\cdot\vec{n}}{|\vec{n}|^2}\cdot\vec{n}-\vec{v}$$

Man könnte nun $I_{spekular}(\vec{k},\vec{n},\vec{s}_L)$ aufstellen durch:

$$\vec{r}\left(\frac{\vec{s}_L-\vec{k}}{|\vec{s}_L-\vec{k}|},\vec{n}\right)\cdot\frac{\vec{v}-\vec{k}}{|\vec{v}-\vec{k}|}$$

Tatsächlich nimmt man aber noch eine n-te Potenz davon, um die durch den spekularen Anteil erzeugten Glanzlichter schärfer zu machen (vgl. Bild 3.16).

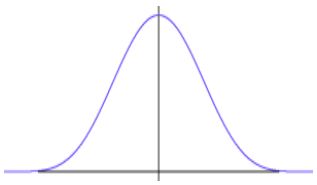

Bild 3.16Graph von $\cos^5 x$

Somit ergibt sich:

$$I_{spekular}(\vec{k},\vec{n},\vec{s}_L)=\left(\vec{r}\left(\frac{\vec{s}_L-\vec{k}}{|\vec{s}_L-\vec{k}|},\vec{n}\right)\cdot\frac{\vec{v}-\vec{k}}{|\vec{v}-\vec{k}|}\right)^n \text{ z.B. mit } n=5$$

Damit ist die Beschreibung dieses grundlegenden Modells fertig. Auf Reflexion, Refraktion und die Fresnel-Gleichungen wird hier aber verzichtet, da diese Eigenschaften nicht unbedingt nötig sind, um den Aufbau eines Raytracers zu begreifen, und die Beschreibung weitaus komplizierter werden würde.

4 Anhang (mit zusätzlichen, von der Korrektur ausgeschlossenen Materialien)

4.1 Einführung in Lineare Algebra

Hier soll das Wissen über Vektoren aufgefrischt werden bzw. auf eine gemeinsame Basis gestellt werden.

Für ausgiebigere Definitionen kann in unserer Formelsammlung[1], in [4] und in [2] nachgeschlagen werden, woraus die folgenden Definitionen (mit kleinen Abänderungen von mir) stammen.

4.1.1 Vektoren und ihre Eigenschaften

1. Ein Vektor besteht aus n reellen Komponenten, man schreibt $\vec{v}\in\mathbb{R}^n$.
2. Es sind folgende Schreibweisen möglich:

   $$\vec{v}=\begin{pmatrix}v_1\\v_2\\v_3\end{pmatrix}=\langle v_1,v_2,v_3\rangle$$

   Letztere Darstellungsweise ist besonders bei komplizierten Formeln nützlich, da sie es erlaubt, den Vektor in nur einer Zeile darzustellen.
3. Vektoren werden komponentenweise addiert und subtrahiert:

   $$\vec{a}\pm\vec{b}=\sum_{i=1}^{n} a_i\pm b_i$$

4. Dabei gilt:

   $\vec{u}+\vec{v}=\vec{v}+\vec{u}$ (Kommutativität)
   $\vec{u}+\vec{v}+\vec{w}=\vec{u}+(\vec{v}+\vec{w})$ (Assoziativität)
5. Sie können skalar multipliziert werden:

   $$a\vec{v}=\vec{v}\,a=a\langle v_1,v_2,v_3\rangle=\langle a\,v_1,a\,v_2,a\,v_3\rangle$$

6. Dabei gilt:

$ab\vec{v}=a(b\vec{v})$ (Assoziativität der skalaren Multiplikation)
$a(\vec{u}+\vec{v})=a\vec{u}+a\vec{v}$ (Distributivität der skalaren Multiplikation I)
$(a+b)\vec{v}=a\vec{v}+b\vec{v}$ (Distributivität der sklaren Multiplikation II)

7. Zwei Vektoren können mittels des Skalarprodukts (auch inneres Produkt genannt) verknüpft werden:
   $\vec{u}\cdot\vec{v}=\vec{u}\,\vec{v}=\langle u_1,u_2,u_3\rangle\cdot\langle v_1,v_2,v_3\rangle=u_1v_1+u_2v_2+u_3v_3=|\vec{u}||\vec{v}|\cos\sphericalangle(\vec{u},\vec{v})$ bzw.
   allgemein: $\vec{u}\,\vec{v}=\sum_{i=1}^{n}u_iv_i$
8. Dabei gilt:
   $\vec{u}\,\vec{v}=\vec{v}\,\vec{u}$ (Kommutativität)
   $a\vec{u}\,\vec{v}=a(\vec{u}\,\vec{v})$
   $\vec{u}(\vec{v}+\vec{w})=\vec{u}\,\vec{v}+\vec{u}\,\vec{w}$
   $\vec{v}\,\vec{v}=\vec{v}^2=|\vec{v}|^2$
   $|\vec{u}\,\vec{v}|=|\vec{u}||\vec{v}|$
9. Zwei Vektoren können über das Kreuzprodukt (auch Vektorprodukt oder äußeres Produkt genannt) verknüpft werden:
   $$\vec{u}\times\vec{v}=\begin{pmatrix}u_2v_3-u_3v_2\\u_3v_1-u_1v_3\\u_1v_2-u_2v_1\end{pmatrix}$$
10. Dabei gilt:
    $|\vec{u}\times\vec{v}|=|\vec{u}||\vec{v}|\sin\sphericalangle(\vec{u},\vec{v})$
    $\vec{u}\times\vec{v}=-\vec{v}\times\vec{u}$
    $a\vec{u}\times\vec{v}=a(\vec{u}\times\vec{v})$
    $\vec{u}\times(\vec{v}+\vec{w})=\vec{u}\times\vec{v}+\vec{u}\times\vec{w}$
    $\vec{v}\times\vec{v}=\vec{0}=\langle 0,0,0\rangle$

4.1.2 Matrizen und ihre Eigenschaften

1. Eine Matrix aus n Zeilen und m Spalten, kurz: $n\times m$, besteht aus $n\cdot m$ Zahlen.
2. Sie wird wie folgt dargestellt:
   $$A=\begin{bmatrix}a_{11}&a_{12}&\cdots&a_{1\mathrm{m}}\\a_{21}&a_{22}&\cdots&a_{2\mathrm{m}}\\\vdots&\vdots&\ddots&\vdots\\a_{n1}&a_{n2}&\cdots&a_{nm}\end{bmatrix}$$
   a_{ij} oder $(A)_{ij}$ bezeichnet dabei die Zahl, die in der i-ten Zeile und j-ten Spalte steht.
   Beispiel:
   $$X=\begin{bmatrix}1&4&5\\3&2&6\end{bmatrix}$$
   $(X)_{23}=x_{23}=6$
3. Eine Matrix heißt **quadratisch**, wenn $m=n$ ist.
4. Eine Matrix A kann **transponiert** werden, gekennzeichnet durch A^T, dabei wird sie an ihrere Diagonalen gespiegelt, mathematisch ausgedrückt:

$$\left(A^T\right)_{ij}=\left(A\right)_{ji}$$

Beispiel:

$$\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}^T=\begin{bmatrix}1 & 3\\ 2 & 4\end{bmatrix}$$

5. Außerdem gilt:

   $$\left(A\,B\right)^T=B^T\,A^T$$

6. Ein Vektor kann als Matrix mit einer Spalte (Spaltenvektor) geschrieben werden:

   $$\vec{v}=\begin{pmatrix}v_1\\ v_2\\ v_3\end{pmatrix}=\begin{bmatrix}v_1\\ v_2\\ v_3\end{bmatrix}$$

   Oder als Zeilenvektor als Matrix mit 1 Zeile:

   $$\vec{v}^T=\begin{bmatrix}v_1 & v_2 & v_3\end{bmatrix}$$

7. Eine Matrix kann auch durch „Untermatrizen" dargestellt werden:

   $$A=\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix};B=\begin{bmatrix}5 & 6\end{bmatrix};C=\begin{bmatrix}7 & 8\end{bmatrix}$$

   $$D=\begin{bmatrix}A & B\\ C & 9\end{bmatrix}=\begin{bmatrix}1 & 2 & 5\\ 3 & 4 & 6\\ 7 & 8 & 9\end{bmatrix}$$

8. Eine Matrix kann folglich auch durch Vektoren dargestellt werden:

   - durch Spaltenvektoren $\vec{a}_1,\cdots,\vec{a}_m$:
     $A=\begin{bmatrix}\vec{a}_1 & \cdots & \vec{a}_m\end{bmatrix}$
   - oder durch Zeilenvektoren $\vec{b}_1^{\,T},\cdots,\vec{b}_m^{\,T}$
     $B=\begin{bmatrix}\vec{b}_1^{\,T}\\ \vdots\\ \vec{b}_m^{\,T}\end{bmatrix}$

   Beispiel:

   $$\vec{a}=\begin{pmatrix}1\\ 3\end{pmatrix};\vec{b}=\begin{pmatrix}2\\ 4\end{pmatrix};A=\begin{bmatrix}\vec{a} & \vec{b}\end{bmatrix}=\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}$$

   $$\vec{c}=\begin{pmatrix}1\\ 2\end{pmatrix};\vec{d}=\begin{pmatrix}3\\ 4\end{pmatrix};B=\begin{bmatrix}\vec{c}^T\\ \vec{d}^T\end{bmatrix}=\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}$$

9. 2 Matrizen werden wie Vektoren komponentenweise addiert und subtrahiert:

   $$C=A\pm B\Leftrightarrow\left(A\pm B\right)_{ij}=\left(A\right)_{ij}\pm\left(B\right)_{ij}$$

   Beispiel:

   $$\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}+\begin{bmatrix}2 & 1\\ 0 & 4\end{bmatrix}=\begin{bmatrix}3 & 3\\ 3 & 8\end{bmatrix}$$

10. Eine Matrix kann mit einer reellen Zahl skalar multipliziert werden:

$$sA = As = \begin{bmatrix} s\,a_{11} & s\,a_{12} & \cdots & s\,a_{1\mathrm{m}} \\ s\,a_{21} & s\,a_{22} & \cdots & s\,a_{2\mathrm{m}} \\ \vdots & \vdots & \ddots & \vdots \\ s\,a_{n1} & s\,a_{n2} & \cdots & s\,a_{nm} \end{bmatrix} \Leftrightarrow (sA)_{ij} = (As)_{ij} = s \cdot (A)_{ij}$$

Beispiel:

$$3\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} 3 & 6 \\ 9 & 12 \end{bmatrix}$$

11. Außerdem gilt für $s, t \in \mathbb{R}$ und A, B als $n \times m$ Matrizen:

$$A + B = B + A$$
$$A + B + C = A + (B + C)$$
$$s(t\,A) = (st)A$$
$$s(A + B) = s\,A + s\,B$$
$$(s + t)A = s\,A + t\,A$$

12. Zwei Matrizen A, B können miteinander multipliziert werden, wenn A eine $n \times p$ und B eine $p \times m$ Matrix ist. Als Ergebnis entsteht eine $n \times m$ Matrix:

$$(AB)_{ij} = \sum_{k=1}^{p} (A)_{ik} (B)_{kj}$$

Anders geschrieben:
Wenn A aus Zeilenvektoren und B aus Zeilenvektoren besteht:

$$A = \begin{bmatrix} \vec{a}_1 \\ \vec{a}_2 \\ \vdots \\ a_n \end{bmatrix}; B = \begin{bmatrix} \vec{b}_1 & \vec{b}_2 & \vec{b}_3 \end{bmatrix}$$

dann ist das Produkt $(A\,B)ij$ der Wert des Skalarprodukts des i -ten Zeilenvektors von A mit dem j -ten Spaltenvektor von B , oder noch einfacher gesagt:
Der Wert der i -ten Zeile und j -ten Spalte der resultierenden Matrix ist das Skalarprodukt der i -Zeile von A mit der j -ten Zeile von B .

Beispiel:

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} 3 & 1 \\ 2 & 4 \end{bmatrix} = \begin{bmatrix} 1 \cdot 3 + 2 \cdot 2 & 1 \cdot 1 + 2 \cdot 4 \\ 3 \cdot 3 + 4 \cdot 2 & 3 \cdot 1 + 4 \cdot 4 \end{bmatrix} = \begin{bmatrix} 7 & 9 \\ 17 & 19 \end{bmatrix}$$

13. Außerdem gilt:

$$sAB = s(AB)$$
$$ABC = A(BC)$$

14. Eine Matrix, deren Diagonale nur aus 1ern besteht und deren andere Felder ansonsten gleich 0 sind, wird allgemein als **Identitätsmatrix** oder **Einheitsmatrix** I bezeichnet.

$$I = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}$$

Dabei wird der Einfachheit halber immer die Matrixgröße angenommen, die zur restlichen Rechnung passt. D.h. wenn man mit 2x2 Matrizen rechnet, so ist mit I die 2x2 Identitätsmatrix bezeichnet, bei 3x3 Matrizen, die 3x3 Identitätsmatrix.

15. Dabei gilt:

    $$AI = IA = A$$

16. Zu manchen Matrizen gibt es eine sogenannte **inverse Matrix**, für welche gilt:

    $$A \cdot A^{-1} = A^{-1} \cdot A = I$$

    Die inverse Matrix von A wird A^{-1} genannt.

    Das Berechnen einer inversen Matrix ist normalerweise recht kompliziert und wird deshalb hier außen vorgelassen, da es einige Vereinfachungen gibt, die der Raytracer benutzen kann. Auf diese wird in der Facharbeit eingegangen.

17. Außerdem gilt:

    $$(AB)^{-1} = B^{-1} A^{-1}$$

18. Zusätzlich kann eine Matrix immer invertiert werden, wenn ihre Transponierte invertiert werden kann.

Anmerkung:
In der Facharbeit werden entweder Zeilen- und Spaltenvektoren (Ersteres aber äußerst selten) oder quadratische Matrizen benutzt.

4.1.3 Matrizen und lineare Gleichungssysteme

Es wird vorausgesetzt, dass Gleichungssysteme bekannt sind. Nehmen wir z.B.:

$$\begin{aligned} a_1 x + a_2 y + a_3 z &= d_1 \\ b_1 x + b_2 y + b_3 z &= d_2 \\ c_1 x + c_2 y + c_3 z &= d_3 \end{aligned}$$

Mit Matrizen kann dieses Gleichungssystem dargestellt werden durch:

$$\begin{bmatrix} a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \\ c_1 & c_2 & c_3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \end{bmatrix}$$

Es lässt sich durch Ausmultiplizieren leicht überprüfen, dass beide Darstellungen äquivalent sind:

$$\begin{bmatrix} a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \\ c_1 & c_2 & c_3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \end{bmatrix} \Leftrightarrow \begin{bmatrix} a_1 x + b_1 y + c_1 z \\ a_2 x + b_2 y + c_2 z \\ a_3 x + b_3 y + c_3 z \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \end{bmatrix} \Leftrightarrow \begin{aligned} a_1 x + b_1 y + c_1 z &= d_1 \\ a_2 x + b_2 y + c_2 z &= d_2 \\ a_3 x + b_3 y + c_3 z &= d_3 \end{aligned}$$

Der Vorteil der Matrixdarstellung liegt darin, dass Koeffizienten, Unbekannte und Lösungen voneinander getrennt sind.

Allgemein lässt sich jedes Gleichungssystem, das die Unbekannten $x_1, x_2, \cdots, x_m$ hat und aus m Gleichungen mit den Koeffizienten $a_{11}, \cdots, a_{1n}, \cdots, a_{n1}, \cdots, a_{nm}$ und den Lösungen $b_1, \cdots, b_m$ besteht, darstellen durch:

$$X = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}; A = \begin{bmatrix} a_{11} & \cdots & a_{1m} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nm} \end{bmatrix}; B = \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix}$$

$$AX = B$$

Durch Multiplikation mit der Inversen, sofern es sie gibt, ergibt sich:

$$AX = B \quad | \cdot A^{-1}$$
$$\Leftrightarrow A^{-1} A X = A^{-1} B$$
$$\Leftrightarrow IX = A^{-1} B$$
$$\Leftrightarrow X = A^{-1} B$$

Und damit lassen sich beliebe Gleichungssysteme einfach lösen.

Damit ist der Grundstein gelegt für die Behandlung von Transformationen und ihrer Verkettung, etc.

4.2 Zusätzliche Beweise

4.2.1 Matrizen aus Basisvektoren sind invertierbar (für Kapitel 3.2.3 Seite 16)

Linear unabhängige Vektoren, also auch Basisvektoren, können keine geschlossene Vektorkette bilden.
Dies wird dadurch ausgedrückt, dass die Gleichung für die Vektoren $\vec{v}_1, \cdots, \vec{v}_2$:
$\lambda_1 \vec{v}_1 + \cdots + \lambda_n \vec{v}_n = \vec{0}$
nur die triviale Lösung $\lambda_k = 0 \,(\text{für alle k})$ hat.

Diese Gleichung ist äquivalent zu:

$$\begin{bmatrix} \vec{v}_1 & \cdots & \vec{v}_n \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \vdots \\ \lambda_n \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \end{bmatrix}$$

Da es genau eine Lösung hat, existiert auch eine Inverse zu $\begin{bmatrix} \vec{v}_1 & \cdots & \vec{v}_n \end{bmatrix}$.
Damit ist der Fall für Matrizen, deren Spalten aus Basisvektoren bestehen, schon bewiesen. Für Matrizen, deren Zeilen aus den Basisvektoren bestehen, kann festgestellt werden, dass ihre Transponierte invertierbar ist (da deren Spalten ja dann aus Basisvektoren bestehen) und mit dem Satz, dass Matrizen genau dann invertiert werden können, wenn ihre Transponierte invertiert werden kann, ist dann auch dieser Fall bewiesen.

4.3 Verwendete Software

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

4.4 Literaturverzeichnis

[1] Barth, Mühlbauer, Nikol, Wörle, "Mathematische Formeln und Definitionen", München, 2003[7], J.Lindauer-Verlag
[2] Barth, Schmid, "Analytische Geometrie (Leistungskurs)", Deutschland, 1985[4], Ehrenwirth
[3] Floating-Point Working Group of the Microprocessor Standards Subcommittee, "IEEE Standard for BinaryFloating-Point Arithmetic", http://754r.ucbtest.org/standards/754.pdf, aufgerufen am 3.10.06
[4] Lengyel, Eric , "Mathematics of 3D Game Programming and Computer Graphics, 2nd Edition", Hingham, USA, 2003, Charles River Media, Inc.
[5] Weisstein, Eric, "Homogeneous Coordinates", http://mathworld.wolfram.com/HomogeneousCoordinates.html, aufgerufen am 6.10.2006

[6] -, "Scene graph", http://en.wikipedia.org/wiki/Scene_graph, aufgerufen am 16.12.2006
[7] -, "Ebene (Mathematik)", http://de.wikipedia.org/wiki/Ebene_(Mathematik), aufgerufen am 1.1.2007
[8] -, "Homogeneous coordinates", http://en.wikipedia.org/wiki/Homogeneous_coordinates, aufgerufen am 30.12.2006
[9] -, "Affine transformation", http://en.wikipedia.org/wiki/Affine_transformation, aufgerufen am 30.12.2006
[10] -, "Transformation (mathematics)", http://en.wikipedia.org/wiki/Transformation_%28mathematics%29, aufgerufen am 18.12.2006
[11] -, "Basis (Vektorraum)", http://de.wikipedia.org/wiki/Basis_(Vektorraum), aufgerufen am 30.12.2006

4.5 Alphabetischer Index

A

Affine Transformation..... **19,** 20f.
Ausgabebild..... 4, 6ff.

B

Basiswechsel..... 14f., 17f.
Bit..... 9
Byte..... 9

D

Diffus..... 26, **27**
Double..... 9f.

G

Grundtransformation.... 5f., 12, 14, 18f., 21

H

Hessesche Normalenform..... **24f.**
Homogen..... **20,** 21ff.

I

Inversen Transformation..... 6

K

Komponentenfunktion..... **11f., 18**
Kontaktpunkt..... 4, 7f., 23ff.
Koordinatenraum..... 5f., 8, 10f., 21, 23

L

Lichtquelle..... 4, 6, 8, 26
Lineare Gleichungssysteme..... **32**
Lineare Transformation..... 12, **13,** 14, 19ff.

N

Normalen..... 7, 16f., 23ff., 27

O

Orthonormal..... **16ff., 23**
Ortsvektor..... 20, 23ff.

P

Primärstrahl..... 4, 6f.
Projektion..... 6ff.
Projektion.....
Orthogonale Projektion..... **8**
Perspektivische Projektion..... **7**
..... **8**

R

Raytracer..... **3,** 4, 6, 8ff., 12ff., 26, 28, 32f.
Rendern..... **4**
Richtungsvektor..... 20, 23
Rotation..... 5f., 12, 15, 17f., 21

S

Sekundärstrahl..... 4, 8
Skalierung..... 5, 12, 15, 18, 21, 23
Spekular..... 26, **27,** 28
Strahl..... 3f., 6ff., 13, 23ff.
Szenengraph..... 4, 6

T

Transformation... 5f., 8, 10, **11,** 12ff., 17ff., 33
Transformationsmatrix..... **13,** 17ff.
Translation..... 5, 12, 19, 21

V

Verkettete Transformation..... 6, 8, 13, 21

Ich versichere, dass ich diese Arbeit selbstständig gefertigt habe; die verwendete Literatur habe ich vollständig angegeben.